APUNTES DE PROGRAMACIÓN EN



“Guia práctica para entender las bondades del R

Fredy Mori Navarro

APUNTES DE PROGRAMACIÓN EN R

F.M.N

20 de Marzo del 2020

Índice general

CAPÍTULO 1

Instalación

1.1 Descargar R

R es un lenguaje de programación dedicado al análisis estadístico y que esta disponible en múltiples sistemas operativos como son Windows, macOS, Linux y hay un desarrollo para android pero que a la actualidad aún no es oficial. Dado el caso de ser un lenguaje de programación, la comunidad ha decidido incluir en su instalador una interfaz llamada "R Commander" que es el estándar pero que a su vez esta pensado ejecutarse en todo tipos de computadoras sin restricción de requisitos(procesador, memoria ram y otros). Debido a su gran fama grupos de desarrolladores han creado IDE'S (entorno de desarrollo integrado) con el fin de que programar en R sea más sencillo y claro que hay otros que se han enfocado en GUI (interfaz gráfica de usuario) centrándose en la creación de menús interactivos similares a IBM SPSS. Dentro de los mas conocidos nos encontramos con:

Rstudio IDE muy recomendado por la comunidad de R pero que para su funcionamiento requiere un procesador dual core como mínimo.

R AnalyticFlow Con un tanto de GUI, como requisito principal es tener instalado Java JDK. Usa diagrama de flujos o nodos, lo que le hace muy conveniente cuando se trata de pronósticos y predicciones donde se necesita repetir el mismo proceso varias veces.

RKWard También con un tanto de GUI y es otra alternativa a Rstudio.

Jamovi, BlueSky Statistics,JASP, GUI muy interactivo para aquellos quienes solo buscan resultados en pocos pasos.

1.1.1 Windows

La descarga de R se encuentra en varios Países mediante la red de servidores cran, comúnmente llamado espejos cran donde almacenan las versiones y documentaciones del software, recuerde que cada servidor o espejo cran es distinto en cuanto a velocidad de descarga.



Para descargar el programa R ingrese a su pagina oficial **https://cran.r-project.org/bin/windows/base/** y haga clic en la versión recomendada.

R-4.0.0 for Windows (32/64 bit)

Download R 4.0.0 for Windows (84 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

Versión Recomendada

Versión Actual Parcheada

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Version en Desarrollo

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

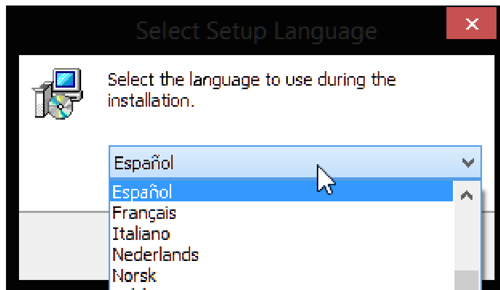
Other builds

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

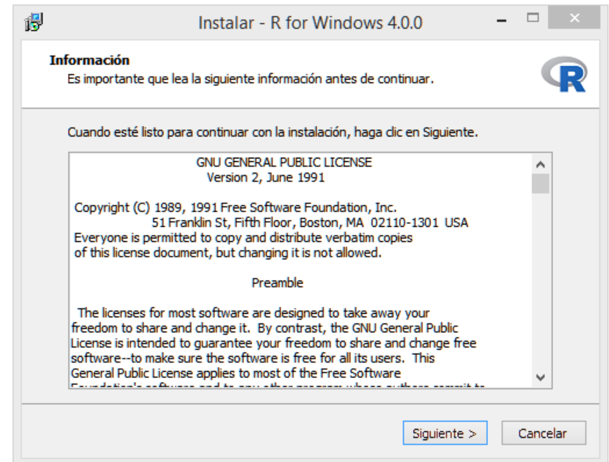
Note to webmasters: A stable link which will redirect to the current Windows binary release is
<CRAN MIRROR>/bin/windows/base/release.htm.

Las otras dos opciones solo son unicamente para probar como va el desarrollo de R, si usted esta en proyectos grandes y no quiere tener errores de software entonces solo use la versión recomendada por la comunidad. Una vez descargado ejecute y haga lo siguiente: “elija las opciones y de en el botón siguiente”

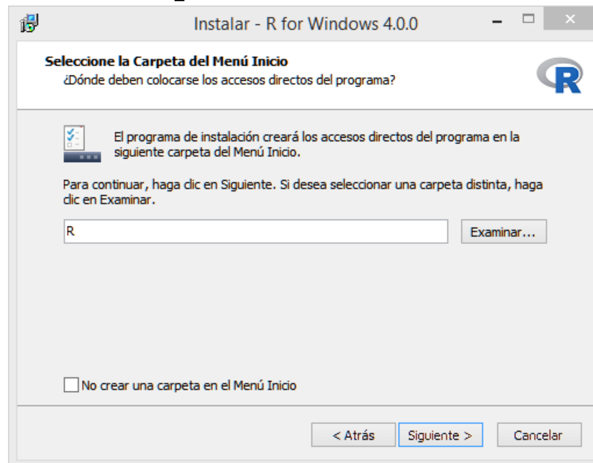
Definir Lenguaje



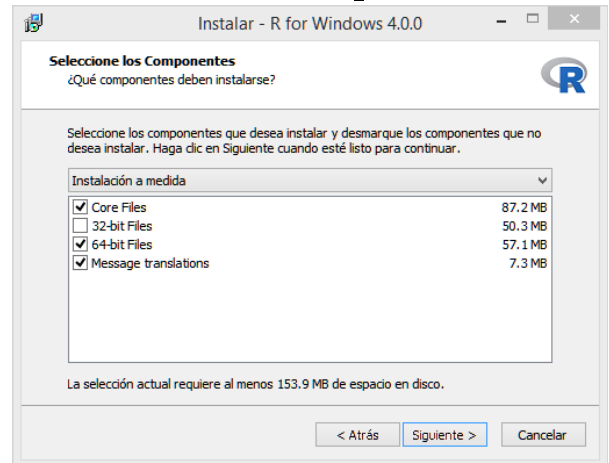
Aceptar licencia



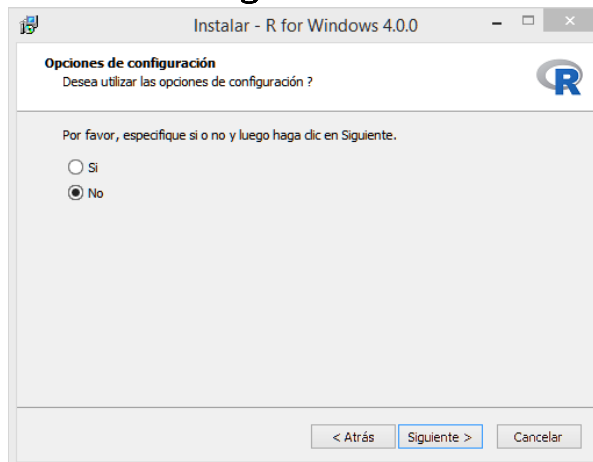
Crear Carpeta en el Menú de Inicio



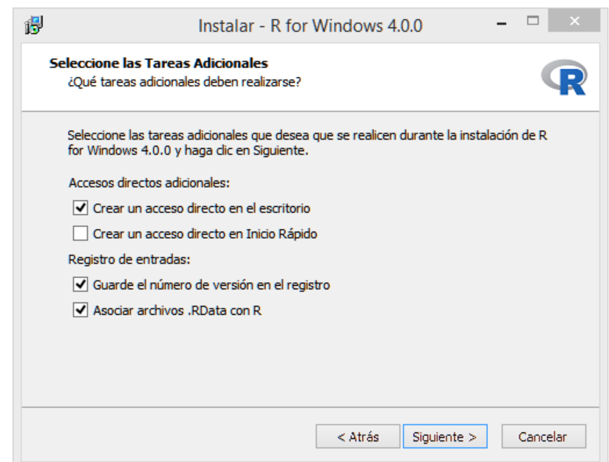
Seleccionar Arquitectura



Usar configuración estándar



Crear acceso directo en escritorio



2.1.2 Mac

La última versión de R unicamente se pueden instalar si cuentas con macOS 10.13(Sierra) y versión superior, Si dispones de versiones de macOS inferiores instala las versiones inferiores de R.

Si posees macOS 10.13, ingrese en esta url directamente a la descarga:
https://cran.r-project.org/bin/macosx/base/
O si tienes una versión de macOS inferior ingrese a:
https://cran.r-project.org/bin/macosx/
y elija la versión que más se adapta a la versión de su macOS.

3.1.3 Distribuciones Linux

Las distribuciones en linux que aceptan la instalación de R-Project son debian,redhat,suse,ubuntu

https://cran.r-project.org/bin/linux/

Debian

Si es la primera vez que instala un software en debian y no posee una clave local "NO_PUBKEY", procure hacer los siguientes pasos.



Paso 1: Abra consola o terminal y escribe

```
sudo nano /etc/apt/sources.list
```

Eso permitira abrir el fichero y agregamos lo siguiente

```
deb http://deb.debian.org/debian/ stretch main contrib
deb-src http://deb.debian.org/debian/ stretch main contrib
deb http://cloud.r-project.org/bin/linux/debian buster-cran40/
```

Estos enlaces servirán para actualizar los componentes base de debian y la última es escencial para R. Una vez terminado presione ctrl+O y ⏎ para guardar los cambios. Actualizamos los repositorios

```
sudo apt-get update
```

Paso 2: Crear clave publica si es que aún no la tienes, en el segundo comando requiere de sus datos, al terminar(la última pregunta), usa V para finalizar y escribe una contraseña para terminar de crear la key.

```
sudo apt-get install gnupg
gpg --full-generate-key
```

Paso 3 Final: escribe este comando en consola y espera a que termine la instalacion

```
sudo apt-get install r-base
```

RedHat

La instalación en Redhat solo requiere de pocos comandos como



```
yum update
yum install epel-release
yum install R
```

El primer comando sirve para actualizar los paquetes, el segundo es un complemento importante para el tercer comando que es el paquete R.

Opensuse

La instalación de R se hace a través de su YaST2 ingresando en
`Main menu -> System -> YaST -> Software -> Software Management`



ejecutas y escribes R-base en su buscador y seleccionas el resultado “R-base” o “R-base-devel”. Finalmente le das clic en el botón Accept y esperas a que empiece la descarga y la instalación.

Ubuntu

Hay dos formas de instalar en ubuntu el software R, una de ellas es mediante `Ubuntu Software` y la otra depende de la consola.



```
sudo apt-get update
sudo apt-get install r-base
```

Si te genera algún error al instalar, es posible que se deba a dependencias del software o a la versión misma del software

"bionic,xenial y focal", puede que este comando solucione tu problema.

```
sudo apt-get install aptitude
sudo aptitude install r-base
```

2.2 Descargar IDE



Importante

Primero debes descargar el software R que incluye el compilador del lenguaje. Después instala el IDE que mejor sea de tu agrado

Como lo mencione al inicio, los IDE'S recomendados por la comunidad son "Rstudio, R AnalyticFlow y RKWard". Cada uno de ellos en sitio web ofrecen la descarga para distintas plataformas.

También hay complementos para editores de código como:

Visual Studio Code R Tools

Atom Rbox

Achitect

Jupyter

Vim Nvim-R

3.3 Softwares Externos

Los reportes Markdown a PDF o LaTeXen Sweave requieren de su propio compilador y es de vital importancia instalar para después no tener errores al compilar un reporte.

El que recomiendo instalar es MiKTeX porque su instalación es limpia sin paquetes a diferencia de Tex Live. Ingrese a esta url para descargar la version reciente "**https://miktex.org/download**", una vez descargado(aprox 235mb) inice la instalación ejecutando este archivo.

Manual Instalación **https://miktex.org/howto/install-miktex**

Nos toca agregar la carpeta a la variable de entorno, **esto es muy importante**. Trate de ubicar el directorio donde se ha instalado MiKTeX, por ejemplo:

```
C:\Program Files\MiKTeX 2.9\miktex\bin\x64
```

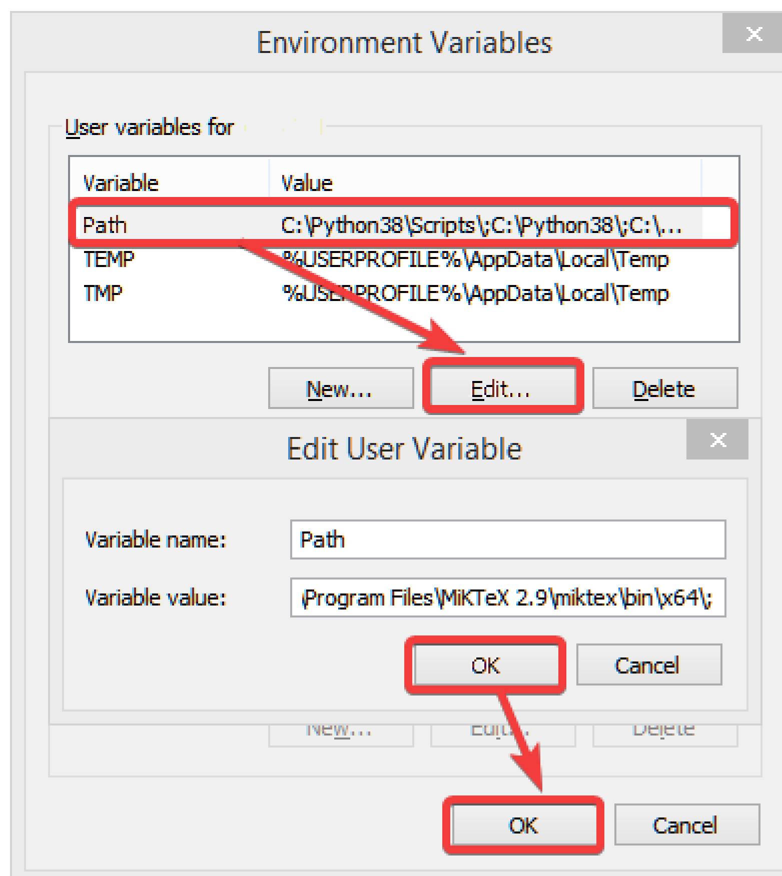
Presione desde windows las teclas [Win] + [R] y escriba

```
%windir%\System32\rundll32.exe sysdm.cpl,EditEnvironmentVariables
```

Esto abrirá una pequeña ventana , enfoquese en **Path** y haga clic en el botón Editar. Al final de ese textbox agrega el directorio

```
;C:\Program Files\MiKTeX 2.9\miktex\bin\x64\;
```

Figura 1.1: *Variable de Entorno Windows*



Los errores comunes en Rmarkdown como es el caso de la siguiente imagen 1.2

Figura 1.2: *Errores de Paquetes faltantes en Rmarkdown y Sweave*

```
Console   R Markdown
.../md/reporte.Rmd
output file: reporte.knit.md

! Sorry, but C:\PROGRA~1\MIKTEX~1.9\miktex\bin\x64\pdflatex.exe did not succeed.

! The log file hopefully contains the information to get MiKTeX going again:

!   C:\Users\       \AppData\Local\MiKTeX\2.9\miktex\log\pdflatex.log

Error: LaTeX failed to compile reporte.tex. See https://yihui.org/tinytex/r/#debugging for deb
ugging tips. See reporte.log for more info.
Execution halted
```

Requieren de ciertos paquetes para compilar el documento, había dicho que Miktex no trae consigo todos los paquetes preinstalados(son alrededor de 6gb), entonces usted desde su computadora ejecute el programa **MiKTeX Console**.

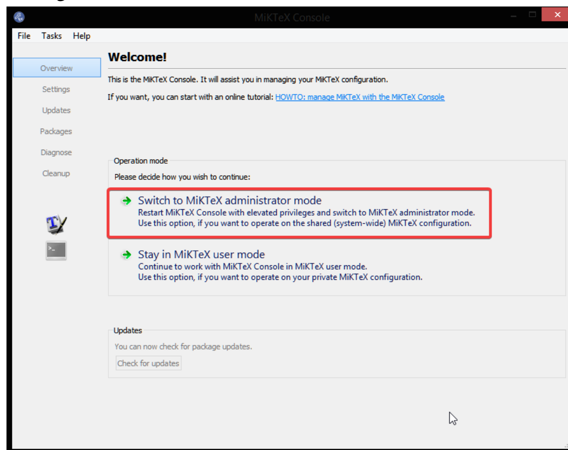
Para Rmarkdown:

- ✔ parskip
- ✔ xurl
- ✔ csquotes
- ✔ footnotehyper
- ✔ biblatex

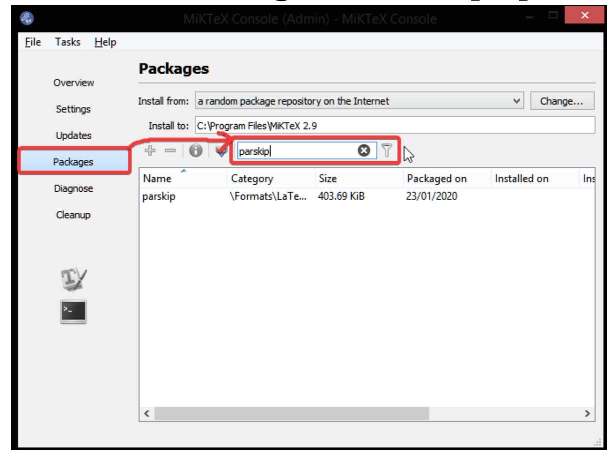
Para Knitr o LaTeX:

- ✔ microtype
- ✔ hyperref
- ✔ layout (*para testear dimensiones de página*)
- ✔ amsmath, amssymb, amsfonts, latexsym, cancel, wasysym
- ✔ array, textcomp, bbding, enumitem
- ✔ ulem, wrapfig, float,pdflscape,tabu, floatrow, subfig
- ✔ longtable,booktabs,multirow,multicol, threeparttable,threeparttablex

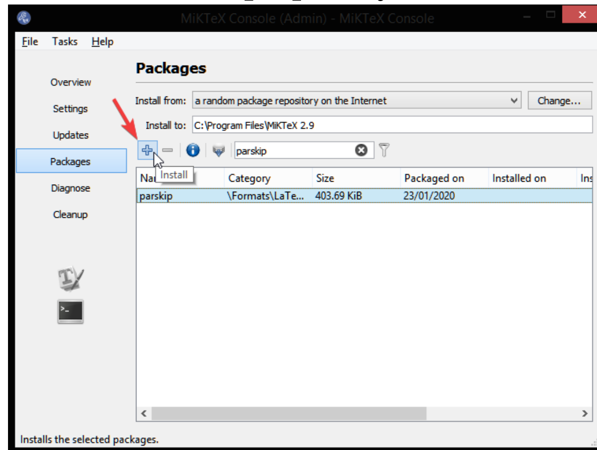
Ejecutar en modo Administrador



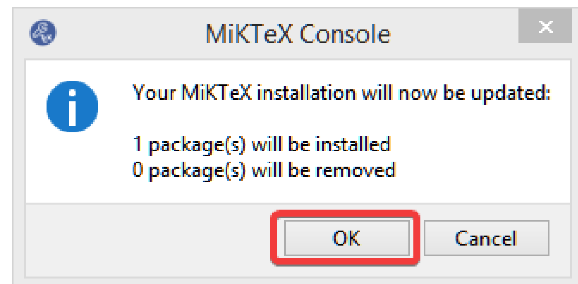
Columna Packages, escribir paquete



Seleccionar paquete y clic en “**+**”



Clic en Ok



CAPÍTULO 2

Objetos en R

1.1 Creando y asignando valores

Para asignar valores a objetos se usa estos símbolos:

$$<-$$

=

Como cualquier otro lenguaje de programación, existe ciertas reglas al crear variables u objetos por ejemplo en el lenguaje R al usar la palabra “**variable**” con “**V**ariable” son distintas así que se considera como otro objeto. El uso de números como nombre del objeto esta prohibido por ejemplo “**12datos,9mos**”, por favor tome nota.

```
> variable <- c(1,2,3)
> variable = c(1,2,3) 1
#--> otra forma 2
>
↪  assign("variable",c(1,2,3))
```



en 1 *se crea un objeto con elementos numéricos.*

en 2 *muestra otra forma de crear el objeto.*

Claro que al nombrar usted puede toparse con nombres de funciones internas como "sd, var, cor, arima, pf y entre otros", si le ha sucedido entonces se habrá dado cuenta que el objeto creado funciona con normalidad pero ello reemplaza a la función y para retornar a su normalidad solo use **rm(nombre_objeto)**. A continuación se presentará algunos casos en donde genera error al nombrar objetos.

Cuadro 2.1: *Nombres de Objetos*

OBJETOS	
✔ Validos	✘ **Inválidos**
`datos <- c()`	`%datos <- c()`
`datos98 <- c()`	`datos% <-c()`
`datos_1 <- c()`	`98datos <- c()`
`a.datos <- c()`	`_datos <- c()`
	`.datos <- c()` [1]

[1] No es específicamente un error(.), sino que no se puede crear el objeto por falta de referencia.

1.1.1 Tipos de datos

En R hay distintos tipos de datos que usted puede crear y dentro de los más conocidos están los siguientes:

Cuadro 2.2: *Tipo de dato en R*

Valores	Tipo
a < − 14	numérico
a < − FALSE	booleano
a < − "b"	carácter
a < − 6**L**	entero "*usar **L** al final*"
a < − 1+6**i**	complejo

Para tratar de saber que tipo de objeto es el que se ha creado, se usará las siguientes funciones:

Cuadro 2.3: *Verificar el objeto*

Función	Especificación
class( )	**C**ategoriza en forma general el objeto y también sirve para interpolar [1]
typeof( )	**D**etermina el tipo interno del objeto, ejem "*NULL,double y otros*"
length( )	**R**etorna la longitud de datos en un objeto
attributes( )	**S**irve para verificar si el objeto tiene algún metadato,objeto complejo
mode( )	**S**imilar a **typeof** pero su clasificación es distinta

[1] Específicamente puede hacer que `class(x) <- c("obj")`, agrega un atributo.

```
> a <- c(FALSE,2,3,5.2)
> class(a)
[1] "numeric"
> typeof(a)
[1] "double"
> length(a)
[1] 4
> attributes(a)
[1] NULL
> mode(a)
[1] "numeric"
```

2.1.2 Clases de objeto

Si se quiere crear un objeto con diferentes tipos de datos "como se muestra en la tabla 2.2", o si son en gran cantidad como por ejemplo big data. Entonces obligatoriamente se tiene que trabajar con una clase que mejor se adapta a nuestras necesidades, a continuación se muestra las clases más usadas:

Cuadro 2.4: *Funciones para crear clase de objetos*

Clase	Especificación
vector()	Crea un vector numérico.
factor()	Usado para codificar un vector en categorías o escalas.
list()	Crea una lista de datos por separado.
matrix()	Crea una $matriz_{nxm}$.
raw()	Crea un vector crudo que almacena longitudes fijas de bytes.
data.frame()[1]	Acepta todo tipo de datos y es muy usado.

En "`vector{mode=" ", length=}`" crea un vector de datos definido por un modo y un tamaño, recuerda que solo genera elementos repetitivos como por ejemplo: `vector{mode="numeric", length=10}` generará dies veces cero. A continuación algunos modos que son aceptados:

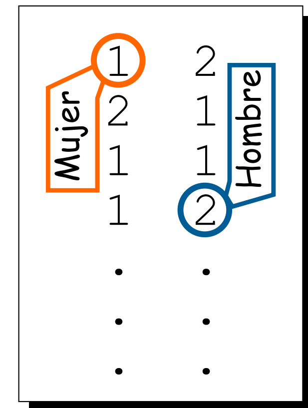
mode ➜ "list, numeric, logical, character,complex, raw"

length ➜ acepta valores numéricos, ejemplo: 10

```
> vector(mode="complex", length=10)
 [1] 0+0i 0+0i 0+0i 0+0i 0+0i 0+0i 0+0i 0+0i 0+0i 0+0i
```

[1]esta clase es muy usado con knitr para crear tablas tipo APA con la librería **kable**, para saber su funcionamiento escribir en consola lo siguiente **?data.frame**, intentar con las otras clases lo mismo **?clase**, cambiar **clase** por "vector,factor,list y otros"

Creando un Objeto **factor**



factor(x ,levels ,labels ,ordered)	
x	(objeto) Necesita un vector de datos cuyos valores coincidan con **levels**.
levels	(numeric) Los niveles o escala de datos.
labels	(string o numérico) La etiquetas a la escala (levels)
ordered	(booleano) Permite ordenar los **levels**.

A continuación se mostrará un ejemplo sobre como funciona:

```
> etiqueta <- c("Nada Satisfecho","Poco Satisfecho",
             "Satisfecho","Muy Satisfecho")
> db <- c(3, 4, 2, 1, 4, 2, 3, 4, 2, 1, 4 ,2,
        1, 4, 1, 1, 1, 4, 1, 4, 1, 1, 1, 4,
        1, 3, 1, 1, 3, 4, 4, 3, 4, 3, 4, 1)

> liker <- factor(x = db ,levels = c(1:4),
                  labels = etiqueta )
```

Ahora si se quiere ver como almacena esta clase **factor**:

```
> liker
 [1] Satisfecho  Muy Satisfecho  Poco Satisfecho Nada
  ↪  Satisfecho
 [5] Muy Satisfecho  Poco Satisfecho  Satisfecho  Muy
  ↪  Satisfecho
  .        .               .               .            .
  .        .               .               .            .
[29] Satisfecho   Muy Satisfecho  Muy Satisfecho  Satisfecho
[33] Muy Satisfecho  Satisfecho  Muy Satisfecho  Nada
 ↪  Satisfecho
Levels: Nada Satisfecho  Poco Satisfecho  Satisfecho  Muy
 ↪  Satisfecho
```

Con un toque estadístico:

```
> summary(liker)      #  --> cantidad
Nada Satisfecho  Poco Satisfecho  Satisfecho  Muy Satisfecho
      14               4               6          12

> prop.table(summary.factor(liker)) # proporcion
Nada Satisfecho  Poco Satisfecho  Satisfecho  muy satisfecho
   0.3888889        0.1111111       0.1666667      0.3333333
```

Errores comunes al crear un factor

Error en levels (1)

```
> liker <- factor(x = l1 ,levels = c(1:3),labels =
 ↪  etiqueta )
--------------------------------------------------------------
Error in factor(x = l1, levels = c(1:3), labels =
 ↪  etiqueta) :
  invalid 'labels'; length 4 should be 1 or 3
```

Error en labels (2)

```
> etiqueta <-
 ↪  c("Nada Satisfecho","Poco Satisfecho","Satisfecho")
> liker <- factor(x = l1 ,levels = c(1:4),labels =
 ↪  etiqueta )
--------------------------------------------------------------
Error in factor(x = l1, levels = c(1:4), labels =
 ↪  etiqueta) :
  invalid 'labels'; length 3 should be 1 or 4
```

En (1) indica que las escalas (1 a 4) de **levels debe ser igual a labels** . Lo mismo sucede en en (2) pero con **labels**.

Creando un Objeto **matrix**

$$\begin{matrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ a_{41} & a_{42} & a_{43} & \cdots & a_{4n} \\ a_{51} & a_{52} & a_{53} & \cdots & a_{5n} \\ a_{61} & a_{62} & a_{63} & \cdots & a_{6n} \\ a_{71} & a_{72} & a_{73} & \cdots & a_{7n} \\ a_{81} & a_{82} & a_{83} & \cdots & a_{8n} \\ a_{91} & a_{92} & a_{93} & \cdots & a_{9n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{matrix}$$

matrix(data ,nrow ,ncol ,byrow, dimnames)	
data	(vector) Necesita un vector de datos cuyos valores coincidan con **levels**.
nrow	(numeric) Numero de filas(**n**) que tiene el vector de datos
ncol	(numeric) Numero de columnas(**m**) que tiene el vector de datos
byrow	(booleano) Se rellena por filas solo si es **TRUE**, caso contrario es columnas.
dimnames	(list) Agrega nombre a filas y columnas, esto depende de la longitud de **ncol** y **nrow**, `dimnames= list(c("filas"),c("columnas") )`.

DATO IMPORTANTE: EL NUMERO DE COLUMNAS (NCOL) Y FILAS (NROW) DEBEN SER SUBMÚLTIPLO DE LA LONGITUD DE TODOS LOS ELEMENTOS, EN CASO DE NO SERLO TE GENERARÁ UNA ADVERTENCIA.

```
> mt <- matrix(data= 1:16, nrow = 4, ncol = 4)
> mt
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

Por cierto si la longitud en filas (nrow) y columnas (ncol) superan la cantidad de elementos que son 16, por ejemplo `ncol = 2, nrow = 9` (9*2 = 18)

```
> mt <- matrix( data = 1:16, nrow = 9, ncol = 2)
```

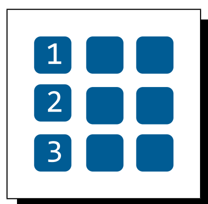
entonces generará una advertencia

```
Warning message:
In matrix(data = 1:16, nrow = 9, ncol = 2) :
  la longitud de los datos [16] no es un
  submúltiplo o múltiplo del número de filas
  [9] en la matriz
```

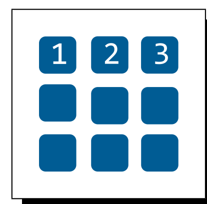
y llenará esa diferencia repitiendo los elementos iniciales como es el caso siguiente:

```
> mt
      [,1] [,2]
 [1,]    1   10
 [2,]    2   11
 [3,]    3   12
 [4,]    4   13
 [5,]    5   14
 [6,]    6   15
 [7,]    7   16
 [8,]    8    1     <-
 [9,]    9    2     <-
```

Ahora se observará como funciona el argumento **byrow** que significa ¿desea ordenar los datos por fila?




```
> mt <- matrix(
        data= 1:9,
        nrow = 3,
        ncol = 3,
        byrow = F)
> mt
     [,1] [,2] [,3]
[1,]  1    4    7
[2,]  2    5    8
[3,]  3    6    9
```




```
> mt <- matrix(
        data= 1:9,
        nrow=3,
        ncol= 3,
        byrow=T)
> mt
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

SE INTENTARÁ AHORA DAR NOMBRE A FILAS Y COLUMNAS. SE DEBE ENTENDER QUE EL **TAMAÑO DE FILAS DEPENDE DE NROW** Y EL **TAMAÑO DE COLUMNAS DE NCOL**

Tomaremos como ejemplo una $\textit{matriz}_{3x2}$

```
> mt <- matrix(c(1:6),nrow=3,ncol=2)
```

dimnames = list(fila, columna)

```
#----------------------------
> dimnames(mt) <- list(
              c("f1","f2","f3"),
              c("c1","c2")      )
#---------------------------
> mt
      c1  c2
f1     1   4
f2     2   5
f3     3   6
```

rownames y colnames

```
#----------------------------
> rownames(mt) <- c("f1","f2",
                        "f3")
> colnames(mt) <- c("c1","c2")
#----------------------------
> mt
      c1  c2
f1     1   4
f2     2   5
```

Creando un Objeto **data.frame**

	data.frame(...,row.names)
...	(objeto) Agregar los objetos creados anteriormente.
row.names	Proporcionar cual columna será tomada como nombres para filas.

Nota: Los tres puntos significa argumentos

El data.frame acepta todo tipo de elementos como son los objetos tipo matrix, factor y vector, pero el único inconveniente es que la longitud de las filas de cada objeto deben ser iguales

```
length(x1) = length(x2) = length(x3) = nrow(b)
```

```
#----------------------------
#    vector
> x1 = c(letters[1:8])
> x2 <- c(1:8)
# Factor
> x3 <- factor(x = c(4,2,4,1,2,1,4,4), levels = c(1:4),
          labels = c("Nada Satisfecho","Poco Satisfecho",
                     "Satisfecho","Muy Satisfecho")    )
#    Matrix
> b <- matrix(data = c(1:16),nrow = 8,ncol = 2)
```

recuerda que `b <- matrix()` *tiene dos columnas.*

row.names = columna 1("lb")

```
#------------
> tb <- data.frame("lb"= x1,
                    "a" = x2,
                    "b" = x3,
                    "c"= b,
                 row.names=1)
> tb # usar View(tb)
  ---------------------------
    a         b        c.1 c.2
  ---------------------------
a   1   Muy Satisfecho   1   9
b   2  Poco Satisfecho   2  10
c   3   Muy Satisfecho   3  11
d   4  Nada Satisfecho   4  12
e   5  Poco Satisfecho   5  13
f   6  Nada Satisfecho   6  14
g   7   Muy Satisfecho   7  15
```

row.names = columna 5("c")

```
> tb <- data.frame("lb"= x1,
                    "a" = x2,
                    "b" = x3,
                    "c"= b,
                 row.names=5)
> tb # usar View(tb)
   ---------------------------
     lb  a         b        c.1
   ---------------------------
9    a   1   Muy Satisfecho   1
10   b   2  Poco Satisfecho   2
11   c   3   Muy Satisfecho   3
12   d   4  Nada Satisfecho   4
13   e   5  Poco Satisfecho   5
14   f   6  Nada Satisfecho   6
15   g   7   Muy Satisfecho   7
```

Errores comunes al crear un data.frame

Error termina en coma “,” 1

```
#-----------------------------
> x1 = c(letters[1:8])
> x2 <- c(1:8)
> tb <- data.frame("lb"= x1,
------------------------------------------------------------


Error in data.frame(lb = x1, a = x2, ) :
```

Error diferentes longitudes de filas 2

```
#------------------------------------
> x1 = c(letters[1:6])
> x2 <- c(1:8)
> tb <- data.frame("lb"= x1,
------------------------------------------------------------
                 "a" = x2,)
Error in data.frame(lb = x1, a = x2) :
```

En 1 es un caso muy conocido tratándose de la coma al final, **se entiende que la coma separa cada argumento de uno nuevo y si no hay uno nuevo entonces este error te generara**.

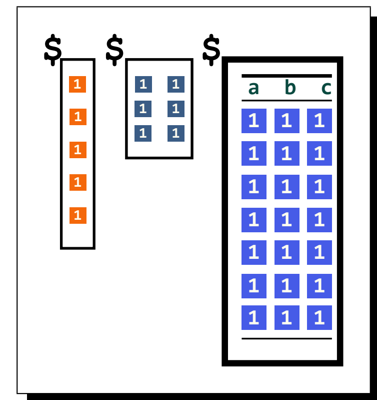
En 2 se trata del tamaño de las filas de cada objeto, como se observa en el primer objeto hay “`length(x1) = 6`” y el otro objeto existe “`length(x2) = 8`” significando distintas filas. **Es muy importante que en data.frame se tenga los objetos con la misma longitud de filas**.

“Uso de `library(tidyverse)`”



Instale este paquete `install.packages("tidyverse")` y usar `tibble()` en vez de data.frame

Creando un Objeto **list**



list(...)	
...	(objeto) Significa que acepta n objetos de distintas clases (matrix, data.frame y otros como funciones) con sus respectivos nombres.

Almacena los objetos en listas independientes, lo de bueno de **list** es que permite agregar clases de objeto como matrix, vector, data.frame, raw y hasta es posible funciones con fórmulas "summary". Además no es importante que dichos objetos tengan la misma longitud en filas y columnas ya que de igual forma serán incluidas.

Por ejemplo:

Se tratará de crear objetos de distintos tipos de clases, como son "vectores, factor, data.frame y matrix"

```
#--------------------------------
> x1 = c(letters[1:8])
> x2 <- c(1:8)
# Factor
> x3 <- factor(x = c(4,2,4,1,2,1,4,4), levels = c(1:4),
         labels = c("Nada Satisfecho","Poco Satisfecho",
         "Satisfecho", "Muy Satisfecho")      )
# data.frame
> tb <- data.frame("lb"= x1, "a" = x2)
# matrix
> mt <- matrix(data= 1:9, nrow=3, ncol= 3, byrow=T)
```

Una vez creado los objetos "observa el último argumento que es una función". Entonces para combinar todo en una lista se tiene que hacer lo siguiente:

```
> lista <- list(x2,x3,tb, mt, summary(x3))
```

Dado que se trata de una lista con múltiples objetos incrustados, entonces únicamente se puede ver a través de consola escribiendo lo siguiente:

```
#---------------------------
> lista
#---------------------------
[[1]]
[1] 1 2 3 4 5 6 7 8

[[2]]
[1] Muy Satisfecho  Poco Satisfecho Muy Satisfecho  Nada
 ↪  Satisfecho
[5] Poco Satisfecho Nada Satisfecho Muy Satisfecho  Muy
 ↪  Satisfecho
4 Levels: Nada Satisfecho Poco Satisfecho ... Muy Satisfecho

[[3]]
  lb a
1  a 1
2  b 2
3  c 3
4  d 4
5  e 5
6  f 6
7  g 7
8  h 8

[[4]]
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9

[[5]]
Nada Satisfecho  Poco Satisfecho   Satisfecho  Muy Satisfecho
      2              2               0              4
```

Observa el caso “**[[5]]**” de **summary(x3)** que también puede ser incluido dentro de la clase **list()**.

3.1.3 Ordenar datos

Los datos en R se ordenan usando el comando **sort()**

```
> datos <- c(5,6,3,4,1)
> ordenado <- sort(datos)
> ordenado
[1] 1 3 4 5 6
```

4.1.4 Valores aleatorios

Hay muchas funciones que generan valores y que estos pueden retornar datos ordenados como también aleatorios pero dentro de los más usados tenemos estas funciones:

Cuadro 2.5: *Funciones para crear valores*

Funcion	¿Que hace?
c(1:n)	Crea una lista de números ordenados que empiezan desde 1 hasta n.
seq(from, to, by)	Genera números desde, hasta y por cuanto aumenta.
rep(x,n)	Repite el dato(no objeto) **x n veces** `Nota: n es un valor numérico`
sequence(nvec)	Genera una secuencia de números muy parecidos a un triángulo.
runif(n, min, max)	Genera números aleatorios con un límite de números “n” que se encuentran entre un mínimo y máximo.
sample(x, size, replace)	Genera números aleatorios de un objeto x pero esta vez cuenta con la opción de hacer que no se repita el mismo número.

```
> v1 <- c(1:5)
> v1
[1] 1 2 3 4 5
#-----------------
```

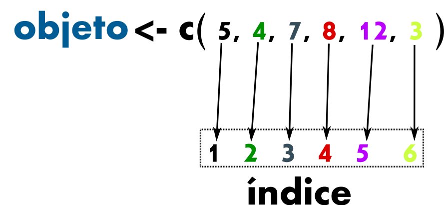
```
> v2 <- seq(1,10,2)
> v2
[1] 1 3 5 7 9
#------------------
> v3 <- rep(letters[2],5)
> v3
[1] "b" "b" "b" "b" "b"
#-----------------
#-----------------
> tmp <- c(4,2)
> v4 <- sequence(tmp)
> v4
[1] 1 2 3 4 1 2
#----------------
#----------------
> v5 <- runif(6,1,5)
> v5
[1] 1.766615 2.726120 1.646968 1.148724 3.180554 2.160788
#----------------
#----------------
> v6 <- sample(1:20,5,replace = T)
> v6
[1] 14 12  5 17  1
```

2.2 Modificar

En las secciones anteriores se ha visto como se crea el objeto y que elementos son permitidos para esa clase de objeto. Las siguientes secciones se hablará sobre la edición o modificación de los elementos del objeto, para ello se ha divido en dos secciones que consta las dimensiones de los objetos.

1.2.1 Índice para objetos de una dimensión

Para poder modificar datos de un objeto es necesario conocer el índice o el orden como es que **R** los almacena



Comenzando por el primer elemento que es 5 su índice sera 1, el elemento 4 su índice 2 y así sucesivamente para los otros elementos. En R se usa dos corchetes que dentro de ello se coloca el índice **[***índice***]**.

```
> datos <- c(5,4,7,8,12,3)
> datos[1]
[1] 5
> datos[3]
[1] 7
> datos[4]
[1] 8
```

Para modificar el valor solamente se agrega el símbolo “$<-$” o el “=” y después el valor nuevo

```
> datos[1] = 12
> datos[3] = 21
> datos
[1] 12 4 21 8 12 3
```

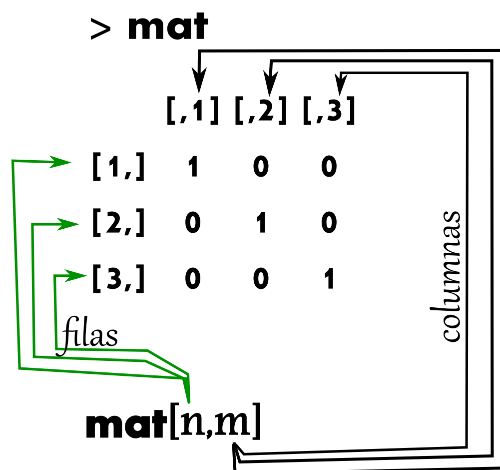
Si lo que desea es modificar múltiples datos entonces use **c(n)**[2] , donde **n** pertenece a los indices a cambiar. Claro que tiene que ser de la misma longitud "**n**:indices y **n**:valores nuevos".

```
> datos[c(1,3)] = c(3,5)
> datos
[1]  3  4  5  8 12  3
> datos[c(2,4,5)] = c(22,24,14)
[1]  3 22  5 24 14  3
```

[2]esto es un vector de datos por ejemplo **c**(1,2,3,4) no confundir con la función **vector()**

2.2.2 Índice para objetos de dos dimensiones

Para ubicar un dato en especifico de una matriz o un data.frame, se entiende que las filas (**n**) son todos los datos en forma horizontal y las columnas (**m**) son todos los datos en vertical, si usamos (n) y (m) entonces apunta hacia un dato específico, por ejemplo:



Se va a tratar de sacar todos los datos en forma diagonal, por favor prestar atención a los indices.

```
> mat[1,1]
[1] 1
> mat[2,2]#fila 2 y columna 2
[1] 1
> mat[3,3]
[1] 1
```

Para cambiar datos solamente se necesita usar “< −”, es muy parecido al proceso que se hizo anteriormente.

```
> mat[1,1] <- 4
> mat[2,2] <- 3
> mat[3,3] <- 5
> mat
     [,1] [,2] [,3]
[1,]    4    0    0
[2,]    0    3    0
[3,]    0    0    5
```

3.2.3 Modificar un data.frame y vectores

Un dataframe no es nada mas que un cuadro de datos que permite almacenar diferentes tipos de datos, aquí hago presente algunas funciones que nos facilitan el trabajo.

Cuadro 2.6: *Funciones que permiten agregar objetos al data.frame*

Función	Especificación
cbind(df,nuevo)	Agrega columnas al data.frame *Nota: tener el mismo tamaño de filas(n).*
rbind(df,nuevo)	Agrega filas al data.frame *Nota: tener el mismo tamaño de columnas (m).*
	retorna la longitud de datos en un objeto
attributes()	sirve para verificar si el objeto tiene algún metadato,objeto complejo
edit(objeto)	Una forma visual de modificar datos *Nota: Es un editor de datos*

```
> cl <- c(2,2,2,2)
> fl <- c(1,1,12)
#------------------
> tb <- data.frame("x"= sample(20:60, 4,replace = F),
                   "y"= round(runif(4,min = 10,max = 20),2))
#-----------------------------
#  Agregar columna
> tb <- cbind(tb,cl)
> tb
    x    y    cl
1  54  12.57  2
2  27  12.72  2
3  48  13.89  2
4  40  10.42  2

#-------------------------
#   Agregar fila
> tb <- rbind(tb,fl)
> tb
    x    y    cl
1  54  12.57  2
2  27  12.72  2
3  48  13.89  2
4  40  10.42  2
5  1   1.00   12

#--------------------------
#   Ver atributos de la tabla
> attributes(tb)
$names                     # solo usa attributes(tb)$names
[1] "x"  "y"  "cl"

$row.names                 # solo usa attributes(tb)$row.names
[1] 1 2 3 4 5

$class                     # solo usa attributes(tb)$class
[1] "data.frame"
```

Si lo que se busca es dar una manipulación mas avanzada en una clase data.frame, entonces debes instalar la siguiente librería que sera de mucha

importancia.

```
> install.packages("data.table")
        ⋮
> library(data.table)
```

Nota: Se necesita internet para la instalación del paquete

Se intentará crear una tabla mixta con el objetivo de tomarlo como referencia.

```
> cl <- rep(2,15)
> fl <- c(1,1,"nada")
> f <- factor(x =
     sample(1:4,15,replace=T),
     levels = c(1:4),
     labels = c("nada",
      "poco","medio","alto"))

> tb <- data.frame(
       "x"= sample(20:60,
        ↪ 15,replace = F),
       "y"= runif(15,min =
        ↪ 10,max = 20),
       "f"= f )

> tb <- cbind(tb,cl)
> tb <- rbind(tb,fl)
#-----------------------------
 Convertir a data.table
#-----------------------------
> tabla <- as.data.table(tb)
```

x	y	f	cl
29	12.39	poco	2
37	11.84	medio	2
31	15.61	medio	2
20	17.31	poco	2
38	10.3	medio	2
43	17.89	poco	2
47	11.65	nada	2
25	19.98	poco	2
46	19.86	medio	2
51	16.61	poco	2
44	12.7	poco	2
52	10.33	alto	2
53	16.16	alto	2
34	14.56	poco	2
48	15.53	nada	2
1	1.00	nada	1

Nota: En la columna "**y**" se ha usado **round(runif( ),2)**

tabla[i, j, by]

Donde:

i	filas (*primer argumento*)
j	columnas (*segundo argumento*)
by	agrupado por (*tercer argumento*)

Extraer datos por filas

Para extraer determinadas filas, en el primer argumento [1:n,] agregar valores numéricos.

```
> tabla[1:4,]
    x     y      f  c1
1: 29 12.39 poco   2
2: 37 11.84 medio  2
3: 31 15.61 medio  2
4: 20 17.31 poco   2
```

También puedes usar una condicional para restringir los elementos que desea ver $[a < n]$ donde **"a" es el nombre de la columna**.

```
> tabla[y<11,]
    x     y      f  c1
1: 38 10.30 medio  2
2: 52 10.33 alto   2
3:  1  1.00 nada   1
```

Si se busca un elemento en específico entonces debe usar **%like%**, o si es el caso de encontrar elementos que se encuentran entre a y b, usar **%between%**.

```
> tabla[x%like%29]
    x     y     f c1
1: 29 12.39 poco   2
#----------
> tabla[y%between%c(11,12)]
    x     y       f c1
1: 37 11.84 medio   2
2: 47 11.65   nada  2
```

Hay una función que permite extraer valores únicos de la tabla por ejemplo la columna "**f**", usar "**unique**(tabla,by)".

```
> unique(x=tabla, by=c("f"))
    x     y       f c1
1: 29 12.39   poco  2
2: 37 11.84 medio   2
3: 47 11.65   nada  2
4: 52 10.33   alto  2
```

Extraer datos por columnas

Para solo ver algunas columnas se necesita ya sea sus nombres como (**x** y **f**) o el número (1 y 3).

```
> tabla[, c(1,3)]
> tabla[, .(x,f)]
     x     f
 1: 29   poco
 .   .    .
 .   .    .
 .   .    .
16:  1   nada
```

La función que permite ordenar columnas es "**setorder**(tabla,-a,b)" por (-,**+**) es (descendente, ascendente).

```
> setorder(x = tabla,x,y )
> tabla
     x     y      f c1
 1:  1  1.00   nada  1
 2: 20 17.31   poco  2
 3: 25 19.98   poco  2
 .   .    .      .   .
 .   .    .      .   .
16: 53 16.16   alto  2
```

Para mas información visite su repositorio en github
https://github.com/Rdatatable/data.table

3.3 Eliminar

1.3.1 Objeto

Si se tiene un objeto similar "**variable**" y se quiere eliminar por completo del sistema con el objetivo de reducir espacio en memoria entonces seria muy oportuno usar la función **rm()** claro que para eliminar múltiples objetos entonces usar coma.

```
> rm(variable)
> variable
Error: objeto 'variable' no encontrado
># ---> eliminar multiples objetos <-------
> rm(a, b, c, mat, df)
> exists("a");exists("b");exists("c");exists("mat");exists("df")
[1] FALSE
[1] FALSE
[1] FALSE
[1] FALSE
[1] FALSE
```

2.3.2 Un elemento o columna

En el tema de modificar se había usado un índice para ver un dato en especifico y bueno en este apartado usaremos el mismo método pero anteponiendo a cada índice un signo negativo indicando la eliminación de dicho dato, por ejemplo:

```
> a <- c(1:4)
> a <- a[-3] #[-n]
> a
[1] 1 2 4
> #eliminar multiple
> a <- a[ -c(1,2)]#-c()
> a
[1] 4
```

Importante: Usar -n para retirar valor con referencia al índice y en cuanto a **-c()**, el signo menos convierte a todos los datos dentro del paréntesis a negativo significando lo mismo **c(-1,-2)**.
Cuando solo se escribe únicamente **a[-3]** imprime en consola los cambios pero no los almacena y por ello el uso de $a < -$.

Data.frame

Es bueno recordar que esta acción no es posible en dataframe:

> **Nota:** no se puede eliminar un dato en específico de una columna dado que las dimensiones serían distintas.

Pero lo que se puede hacer es eliminar filas completas y columnas, preste atención a este código

```
> dt = data.frame(
      "h1"=c(runif(5,40,64)),
      "h2"=c(sample(2:25,size = 5))
      )

> dt[-c(1,4),] # eliminar n = -1, -4
      h1    h2
2 53.59845 18
3 46.13610 16
5 55.03759 19
> dt <- dt[-c(1,4),] # almacenar en dt
```

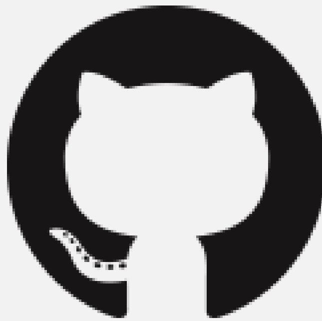
		h1	h2
X	**1**	50.43018	15
	2	53.59845	18
	3	46.13610	16
X	**4**	61.00323	11
	5	55.03759	19

Para seleccionar una columna se necesita usar el símbolo “$”, por ejemplo **nombre$columna** le decimos que sea igual a una acción llamada “**NULL**”.

```
> dt$h1 <- NULL
```

	h1 (X)	h2
2	53.59845	18
3	46.13610	16
5	55.03759	19

Descarga de Códigos



Puede usted visitar el repositorio github y encontrará todos los comandos escritos en este capitulo, recuerde visitar en este enlace **https://github.com/Moriand/Apuntes-R**

CAPÍTULO 3

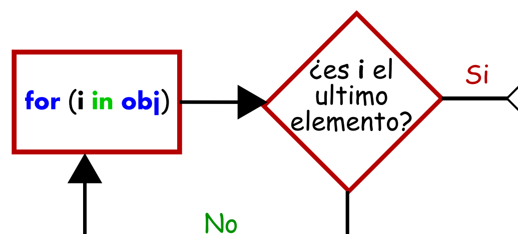
Repetidores y Condicionales

1.1 Repetidor

1.1.1 For

El repetidor **For** crea un ciclo de elementos de un objeto establecido **for (variable in obj)**. El **in** indica de que objeto (**obj**) se extraerá los elementos y los almacena en **variable**.

```
> obj <- sample(40:95,size
  ↪   = 10)
> for (i in obj){
        print(i)
  }
```



Datos a tener encuenta:

- ✔ No necesariamente debe llamarse **i** por ejemplo: salida,tmp y otros.
- ✔ El objeto a repetir debe tener una **longitud mínima de un dato** c(dato).
- ✔ Es importante que este entre paréntesis “**()**”.

Para obtener cada elemento de una matriz o un data.frame entonces se usará un doble bucle For

```
> dt = data.frame("x"=c(sample(1:15,size = 5)),
                  "y"=c(sample(2:25,size = 5)),
```

```
"z" = sample(30:65,size = 5),
row.names =c("a","b","c","d","e") )
```

```
#-a-- por columna
> for (i in 1:ncol(dt)){ 1
    for (j in 1:nrow(dt)){ 2
     print(dt[j,i])
    }
  }
```

```
#-b-- por fila
> for(i in 1:nrow(dt)){
    for(j in 1:ncol(dt)){
      print(dt[i,j])
    }
  }
```

```
#-c-- por nombres fila
> for (i in rownames(dt)){ 3
    print(i)
  }
```

```
#-d--todos
> for (i in dt){
    print(lb)
  }
```

Tips en código

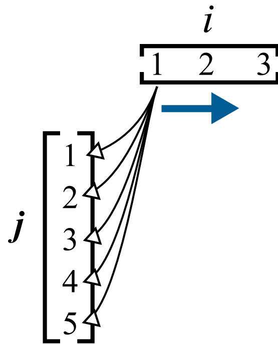
En 1 ncol(dt) = *dim*(*dt*)[2] y 2 también nrow(dt) = *dim*(*dt*)[1], por si quieres evitar el **1:** entonces aquí tienes una alternativa 1:nrow(dt) = *seq_len*(*dim*(*dt*)[1])

		1	2	3
		x	y	z
1	a	6	18	63
2	b	9	14	58
3	c	13	5	45
4	d	8	13	43
5	e	11	11	46

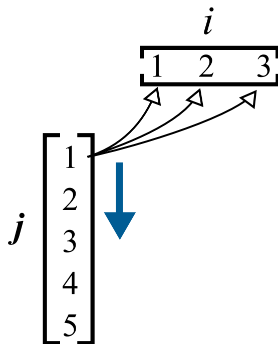
1 2 3

1
2
3
4
5

a el **primer for** genera números de 1 hasta m columnas y el **segundo for** genera desde 1 hasta n filas:



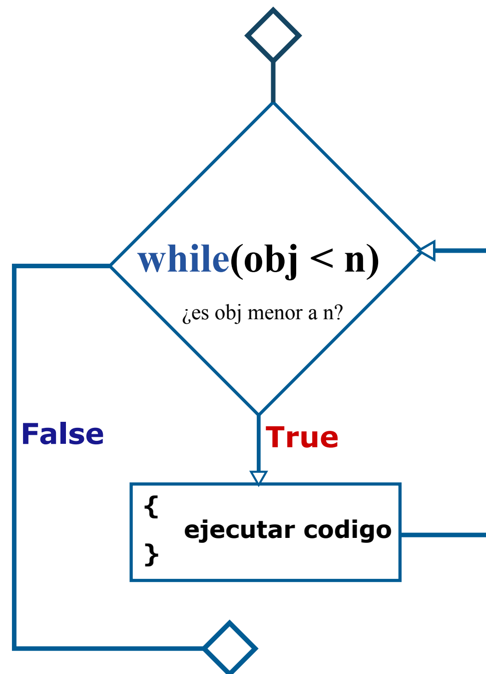
b el **primer for** genera números de 1 hasta n filas y el **segundo for** genera desde 1 hasta m columnas:



c Aquí solamente extrae los nombres por filas y seguramente ya los conoces.

d En esta parte el **for** extrae todos los elementos del objeto **dt**, no confundir con los otros que solamente generan el índice. Es importante aclarar que genera una longitud > 1 y no es posible usar if en ello.

2.1.2 While



La función while es un repetidor con una condición que mientras sea cierta entonces se ejecutará todo el pedazo de código que se encuentre dentro, por ejemplo tengo un objeto

```
> obj <- c(1:10)
```

después creo un **while** con la condición mientras obj sea **menor de** 10 entonces se ejecutará el otro código donde imprima "hola mundo", ten cuidado con este tipo de condiciones que hace repeticiones infinitas dado a que siempre retornara un **TRUE**

```
> a = 1
> while(a < 8){
     print( paste("hola", a, sep = "->") )
     a = a+1
  }
```

Retornará "hola$->$i" donde $i \epsilon \{1, \cdots 7\}$, así que se repetirá siete veces por el motivo de la condición.

3.1.3 Repeat

Es un repetidor sin condiciones, así que se tiene que colocar una condición para que finalize y se usará un **break** para que finalize **repeat{ }**, a tener mucho cuidado al usar esta función

Ejemplo repeat

```
re = 0
> repeat{
    re = re + 1
    if (re == 10){
      break            1
    }else{
      print("hola")
    }

  }
```

En 1 el break rompe la función repetidor y hace que finalice. Con respecto a **if** por favor revisa la sección **If**

La función repeat crea un ciclo ilimitado de repeticiones y solamente parará si se controla con una condición.

El caso del objeto “re” solo nos sirve como contador y por defecto el valor principal es “0”, dentro de la función repeat observe que “re” se suma el valor actual del mismo con una unidad, o sea (0+1,1+2,3+1,...). En la condición ha sido colocado un valor máximo de 10 repeticiones que “re” puede tomar.

2.2 Condicional

Antes de empezar, usted debe conocer los operadores lógicos en su totalidad y para ello se ha preparado las siguientes tablas donde indican cuales son los símbolos y como funcionan

Cuadro 3.1: *Símbolos usados para comprobar elementos*

Operador lógico	¿Que hace?
!	Niega a toda la falacia (!p == q) retornando un valor lógico contrario.
&	El "**and**" retorna valores lógicos para cada elemento condicionado. [1]
&&	Solamente retorna un solo valor lógico, es True cuando solo ambos elementos lo son
|	El "**or**" retornar valores lógicos por cada elemento
||	Solamente retorna un solo valor lógico y es FALSE cuando ambos elementos se contradicen.

Figura 3.1: *Función con varios operadores lógicos*

```
for (i in a){
   for (j in b){
     if( (i == j) && (i%%2 == 0) ){
       }
   }
}
```


Cuadro 3.2: *Operadores Lógicos*

p	q	&	|	xor(p,q)
TRUE	**TRUE**	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	TRUE
FALSE	**TRUE**	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE

[1] por ejemplo `a = c(1:8)` y uso la siguiente condición `a>2 & a<8`, esto retorna ocho valores lógicos. Pero si hago lo siguiente `a>2 && a<8`, entonces retorna un solo valor lógico.

Cuadro 3.3: *Símbolos usados para comprobar o relacionar elementos*

Símbolo	¿Que hace?
a[i] == **b[j]**	sirve para verificar si **a** es igual que **b**
a[i] <= **b[j]**	verifica si **a** es menor o igual que **b**
a[i] >= **b[j]**	verifica si **a** es mayor o igual que **b**
a[i] != b[j]	verifica si **a** es diferente de **b**
a[i] %in% **b**	verifica si el elemento **a[i]** se encuentra en el objeto **b**

Cuadro 3.4: *Verificar y Convertir*

Verificar	Convertir
is.vector(x,mode)	as.vector(x,mode)
is.factor(x)	as.factor(x)
is.list(x)	as.list(x)
is.matrix(x)	as.matrix(x)
is.raw(x)	as.raw(x)
is.data.frame(x)	as.data.frame(x, row.names)
is.na(x)	
is.nan(x)	
is.logical(x)	as.logical(x)
is.numeric.Date(x)	as.Date.character(x)

1.2.1 If

En esta sección la sentencia "if" cuestiona si se ejecuta o no una parte del código que esta dentro de llaves "{ ··· }" sin embargo en el lenguaje R nos ofrece dos formas de usar la sentencia **if**.

Modo Reducido

Contamos con una función sencilla llamada
ifelse(comprobar,retorna1,retorna2)
en comprobar se tendrá que colocar el (objeto más la condición) que se desea verificar y si esto es verdad(TRUE) entonces **retorna1** caso contrario retorna2. Por ejemplo:

```
> a = c(runif(5,40,64))
[1] 60.36799 52.96228 48.25127 42.66303 44.14928
> ifelse(a>45,"si","no")
[1] "si" "si" "si" "no" "no"
```

Comprobar si hay números pares

```
> a = round(runif(5,2,10),0)   1
> a
[1] 10  5  9  5  5
> b = c(2,4,6,8)
> c = c(1,3,5,7)
> ifelse(a%%2==0,b,c)   2
[1] 2 3 5 7 1
```

En 1 primero uso la función round(objeto,redondear) para redondear valores por el motivo que runnif genera valores con decimales.
En 2 verifico si cada elemento de **a** es par y uso un módulo (%%2== 0) que se encarga de ello.

Si quieres que sea mas eficiente con respecto a velocidad de procesado, entonces usa lo siguiente:

```
> dplyr::if_else(condicion,si,no,missing=3)
```

Funciona igual que el otro con la diferencia que en **si** y **no** tiene que ser el mismo tipo de datos y la misma longitud. En missing hace referencia a valores nulos (NA) y permite cambiarlos por otros valores como es el caso de 3 (**no compara**).

```
> require(dplyr)
> a = c(10,5,9,5,NA) # n=5
> b = c(2,4,6,8,10) #  n=5 --> numeric
> c = c(1,3,5,7,9)  #  n=5 --> numeric
> dplyr::if_else(a%%2==0,b,c,missing=11)
[1] 2  3  5  7 11
```

Modo Completo

Es probable que sea el más usado en programación ya que permite agregar código dentro del paréntesis y esto es muy bueno por el motivo que separa entre cual se ejecuta y cual no. El if en español “si”se entiende como: Si el objeto es menor que 8, **entonces** se ejecuta print e imprime “trozo de código a”, **caso contrario** pasa a ejecutarse print e imprime “trozo b”.

```
> if (objeto < 8){
    # si es TRUE
    print("trozo de código a")
```

```
}else{
  #si es FALSE
  print("trozo de código b")
}
```

Ejemplo de If

```
> a = sample(2:20,5, replace=F)
> b = sample(2:20,5, replace=T)
> for (i in a){
     for (j in b){
       if ((j == i) & (j%%2 == 0)){ 1
           lb = "es par e igual a j"
           print(paste(j,lb,sep=" -> "))
       }
     }
  }
```

Se interpreta como: Si ((¿ el elemento j del objeto b es igual al elemento i del objeto a?) **y** (¿ el elemento j es par?)).
Nota: los valores a y b son generados aleatoriamente, así que intente usted desde su editor preferido.

Dato Importante

Por si genera un error como esto:

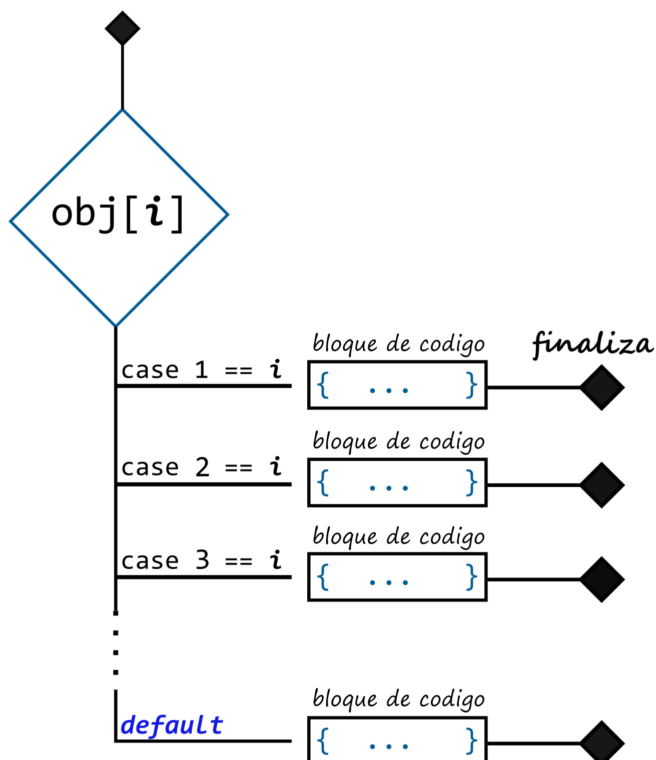
the condition has length >1 and only the first element will be used

Recuerda que el condicional **if** solo compara un elemento por objeto por ejemplo df[elemento] == 2. Si deseas multiples resultados y sin un for entonces usa **ifelse**

2.2.2 Switch

switch(expresion, case 1, case 2, case 3...,default)

Figura 3.2: *Diagrama de flujo switch*



Cuando usted se encuentra en un mercado y observa que hay n marcas del mismo producto que desea y por lo general ofrecen el mismo resultado pero usted quiere una marca en especifico, entonces se acerca al vendedor para preguntar por dicha marca.

El funcionamiento de switch es lo mismo por el motivo de que los casos (case n) con el cual se compara ya están dadas y solamente se esperaría escribir con que elemento (expresión u objeto) se desearía encontrar una igualdad.

Ejemplo del supermercado

```
> marca <- readline("ingrese marca > ")   1
> switch(marca,
    "ayc"=print(paste("disponible, marca",marca,sep=" -> ")),
    "sold"=print(paste("disponible, marca",marca,sep=" -> ")),
    "lilo"=print(paste("disponible, marca",marca,sep=" -> ")),
    "mrc"=print(paste("disponible, marca",marca,sep=" -> ")),
```

```
    print("no disponible")  ❷
  )
```

En ❶ es una entrada de datos que manualmente usted tiene que escribir, puede recibir error si hace un mal uso de esta función.
En ❷ se conoce como **default** indicando que si las demás no se cumple entonces ésta se ejecuta.

Nota importante

No puedes condicionar objetos que tenga n elementos (superior a un elemento), es posible que encuentres este error:

```
Error in switch(names(dt), x = print("cierto")) :
  %EXPR must be a length 1 vector
```

No puedes terminar en coma cuando estés en el ultimo case, este error te generará:

```
Error: duplicate 'switch' defaults: 'print("no ...' and ''
```

por ejemplo en el ❷ esta linea no termina en coma y no te va a generar error.

3.2.3 Which

which(x, arr.ind, useNames)

x	Es el objeto con el operador lógico.
arr.ind	Si es TRUE entonces el objeto es una matriz caso contrario un vector común.
useNames	Si el objeto es una matriz, entonces al hacer useNames=TRUE muestra los nombre de filas, caso contrario mostrará números.

Esta función solo retorna el índice del elemento que se encuentran en dicha condición y esta dada dentro del paréntesis **which(condición)**. Dicho esto which significa **cual** y en español seria por ejemplo: ¿cual elemento del obj es igual a 3? which(obj==3).

Ejemplo which con vector normal

```
> a <- c(2,5,4,2,4)
> which(a==2)
[1] 1 4
```

El resultado es el índice o sea a[1] = 2 y a[4] demostrando que son igual a 2.**a[indice]**

Ejemplo which con matriz

```
> mat <- matrix(data = c(
    4, 5, 0,
    5, 8, 0,
    3, 6, 1
  ),
  nrow = 3, ncol = 3, byrow = TRUE)
> rownames(mat) = c("c1","c2","c3")
> colnames(mat) = c("f1","f2","f3")
> which(mat<2,useNames = T,arr.ind = T)
    row col
c1   1   3
c2   2   3
c3   3   3
```

Se interpreta como mat[1,3] = 0 y esto demuestra ser menor que 2. **mat[row,col]**

Descarga de Códigos



Puede usted visitar el repositorio github y encontrará todos los comandos escritos en este capitulo, recuerde visitar en este enlace **https://github.com/Moriand/Apuntes-R**

CAPÍTULO 4

Atrapando Errores

En muchas ocasiones usted al intentar ejecutar una función o un script se ha topado directamente con muchos errores y que en cuestión le han imposibilitado ejecutar correctamente todo el código, pero por suerte casi en su gran mayoría de los paquetes siempre tienen esta función y es de vital importancia saber como es su funcionamiento y en que nos puede beneficiar.

1.1 try

Significa "intentar" y ofrece solo una opción que es "*silent*" siendo un booleano. Esto permite omitir los mensajes que pueden aparecer si la expresión escrita "···"genera algún tipo de alerta.

```
try(...,silent=TRUE)
```

Por ejemplo si tengo un vector de datos y a esto quiero sumarlo con un carácter de la siguiente manera:

```
> mat <- c(1,4,2,6,4)

> try(mat+"a",silent=FALSE)
Error in mat + "a" : argumento no-numérico para operador
 ↪  binario
```

Obtenemos un mensaje, entonces si aplico `silent=TRUE`:

```
> try(mat+"a",silent=TRUE)
>
```

Omite el error y continua con la siguiente linea.

Le resultaría muy útil al tratar de omitir mensajes al hacer operaciones sencillas como por ejemplo: sacar la inversa de matrices, multiplicar matrices y entre otros.

2.2 tryCatch

La palabra tryCatch hace referencia a intentar ejecutar el bloque de código y si tiene error, lo captura y nos devolverá según se halla programado:

y bueno he aquí la pregunta.

¿Cuantos niveles hay?

Según la función tryCatch considera dos tipos de niveles de errores

warning (leve) El tipo de mensaje que aparece como alarma pero en si hay un funcionamiento(retorna valores), por ejemplo al crear una matriz con numero total de elementos no son submúltiplos entre numeros de columnas.

error (grave) En este nivel el comando o función no llega a ejecutarse(no retorna valores) y pasa a ser anulado por el mismo R.

```
tryCatch(
    expr = {
        # Operaciones...
        # aquí...
    },
    error = function(e){
        # si hay error
        # ejecutar aquí
    },
    warning = function(w){
        # si hay alarmas
```

```
        # ejecutar aquí
    },
    finally = {
        # (Opcional)
        # se ejecutará como final
    }
)
```

Tenemos aqui un ejemplo práctico.

```
> matrices <- function(matriz_1,matriz_2){
    tryCatch(
      expr = {
        matriz_1 %*% matriz_2
      },
      error = function(e){
        message('--- error de dimensión ---\n')
        print(e)
      },
      warning = function(w){
        message('------------ alarma ------\n')
        print(w)
      },
      finally = {
       message('----- final -----------------')
      }
    )
  }
#===========================

> a <- matrix(data = c(2,3,6,
                       3,8,7),nrow = 3,ncol = 3)

> b <- matrix(data = c(2,3,6,3,
                       8,7,4,6),nrow = 2,ncol = 4)
> #--------------
> matrices(matriz_1 = a,matriz_2 = b)
------- error de dimensión ----- ❶

<simpleError in matriz_1 %*% matriz_2: argumentos no
 ↪   compatibles> ❷
----- final ----------------- ❸
```

Primeramente es un **error** y es por eso que en ❶ imprime “message”, después pasa a **finally** imprimiendo también “message”.

Si el error hubiera sido leve, pasaría a ejecutarse “warning”.

Capítulo 5

Funciones en R

1.1 Funciones sin argumentos

Son las funciones sencillas de crear ya que no requieren de objetos externos(argumentos) para funcionar correctamente, por ejemplo este es la estructura de una función simple:

```
> clase <- function(){
    # codigo -->
  }
```

La palabra **clase** es el nombre la función, debes tener cuidado al nombrar ya que por casualidad puedes toparte con nombres de funciones internas y esto crea una función temporal que reemplazará por el creado (cierra el programa o elimina la función con **rm**(nombre_funcion) y volverá a la normalidad). Como son en gran cantidad las funciones internas no se los nombrará pero si los más importantes "if, try,for, View,rm".

Dato Importante

Este es el orden para poder usar una función correctamente:

1) Crear función "`nombre_funcion = function(){ }`"

2) Llamar `> nombre_funcion()`

También debes **evitar usar caracteres con tilde** en el nombre. Al llamar debes **usar los paréntesis** después del nombre sino lo haces estarías imprimiendo la estructura de la función.

Si por casualidad quieres saber la estructura de la función, entonces simplemente escribe en consola lo siguiente `> nombre_funcion` y listo. Para obtener el valor de la función debes usar los paréntesis al final del nombre, por ejemplo `> nombre_funcion()`.

Observe este ejemplo que muestra de una forma más detalla como funciona:

```
> Nombre <- function(){
      print("Hola Mundo")
  }
> Nombre()          1
[1] Hola Mundo
> Nombre            2
function(){
  print("Hola Mundo")
}
```

Por cierto también hay ocasiones en donde se comete errores al tratar de crear funciones y dentro de los más cotidianos se encuentran los siguientes.

LOS DE LA IZQUIERDA SON LOS ERRORES Y DERECHA ES COMO CORREGIRLO:

ALGUNOS ERRORES COMUNES



```
> 1er <- function(){
    print("error nombre")
  }
```



```
> er1 <- function(){
    print("nombre")
  }
```



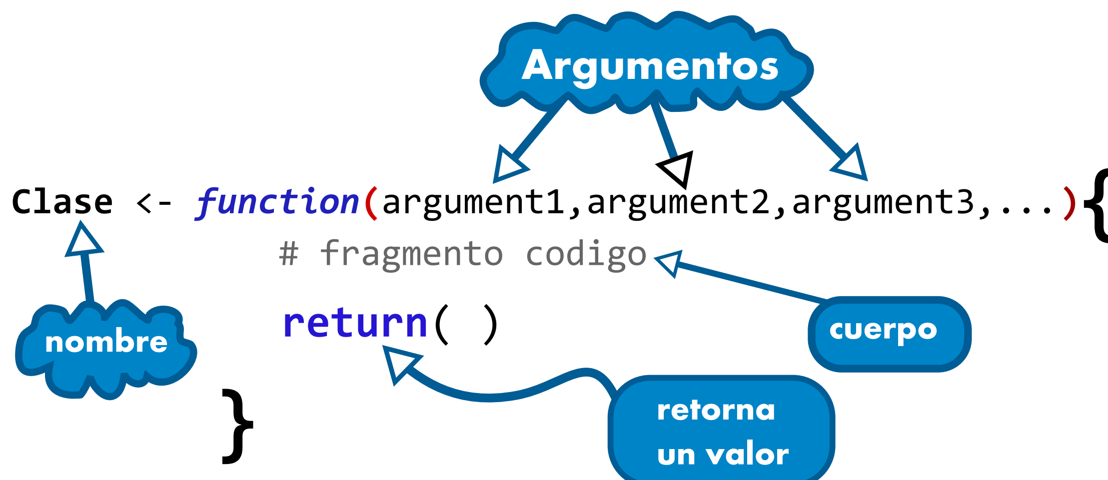
```
> primer <- function{
  print("error ()")
  }
```



```
> primer <- function(){
    print("parentesis")
  }
```

Recuerda que si hay un error en alguna parte de la función o en el contenido del código como `print()`, esto no ejecuta el bloque de código en su totalidad sino mas bien te imprimirá un texto de color rojo indicando parte del error y algunas pistas sobre donde esta el error generado.

2.2 Funciones con argumentos



Los argumentos no son mas que referencias que requiere la función para su funcionamiento, se debe entender que estas referencias son objetos y que solo estarán disponible dentro de la función. También se cuenta con un **return()**, ésto sirve para hacer un retorno de solo un objeto o valor específico en toda la función.

```
# funcion simple
> Clase <- function(a){
      tmp = log(a)/2
      return(tmp)
  }
```

Sin Referencia

```
> Clase()
Error in Clase() :
 el argumento "a" está
  ↪  ausente,
 sin valor por omisión
```

Con Referencia

```
> Clase(a = 5)
[1] 0.804719
> Clase(a = 0)
[1] -Inf
```

Como se observa en la función **Clase**, tiene un objeto llamado **a** o bien puede llamarse cualquier otro nombre, lo que debes entender es que este objeto solo estará disponible dentro de la función y no fuera, ejemplo:

```
Error: objeto 'a' no encontrado.
```

Ahora el objeto **a** obligatoriamente tiene que recibir una referencia, caso contrario genera un error.

Por ejemplo: se creará una función con dos argumentos "**valor** e **intentos**" donde **valor** recibe un número que se encuentre entre 1 y 10 e **intentos** igual, dentro de la funcion (cuerpo) se tratará de crear un objeto temporal que genera números aleatorios con un tamaño dado en el objeto **intentos**. Despues se creará un for para verificar si el objeto temporal es igual al **valor**.

```
#---- funcion probar suerte
> suerte <- function(valor, intentos){
    tmp = sample(1:10,size = intentos, replace = F)
    for (i in seq_along(tmp)){
      if (tmp[i] == valor ){
        return(print(" --> valor encontrado") )
      }
    }
    return(print("intentelo nuevamente"))
  }
```

Por casualidad intenté con estos valores y he acertado, trate usted dar valores numéricos:

```
> suerte(valor=2, intentos=4)
[1] "intentelo nuevamente"
```

```
> suerte(valor=6, intentos=4)
[1] " --> valor encontrado"
```

¿ *Que pasa si escribo mal la función*?

Errores comunes

Coma sin argumento (1)

```
> suerte <- function(valor, intentos,){
```

```
Error: inesperado ')' in
↪  "suerte <- function(valor, intentos,)"
```

Omitir un paréntesis (2)

```
for (i in seq_along(tmp){
...
}
```

```
Error: inesperado '{' in:
"  tmp = sample(1:10,size = intentos, replace = F)
  for (i in seq_along(tmp){"
```

En (1) elimine la última coma y en (2) agrege el parénte que encierra a **for ()** .

Creando un convertidor: se creará una función **convert** con dos argumentos "**dt** y **tipo**" donde:

- ✔ **dt** Hace referencia al objeto que tiene los datos.
- ✔ **tipo** Hace referencia a que tipo de objeto deseas convertir.

Como primer paso se necesita verificar si el objeto a convertir no se encuentra vacío

- ✔ **Objeto vacío** Imprimirá en pantalla una alerta detallada y no logrará convertir.
- ✔ **Objeto no vacío** Usando **switch()** que depende del **tipo** se procederá a convertir dicho objeto.

```
#----> multiconvertidor
> convert <- function(dt, tipo="factor"){
    if (! is.null(dt)){
        switch(tipo,
            "vector"= {
                return(as.vector(dt))
            },
            "factor"= {
                return(as.factor(dt))
            },
            "list"= {
                return(as.list(dt))
            },
            "matrix"= {
                return(as.matrix(dt))
            },
            "raw"= {
                return(as.raw(dt))
            },
            "data.frame"= {
                return(as.data.frame(dt))
            },
            print("tipo no existe")

           )

    }else{
        warning("Verifique si no esta vacio")
    }
  }
```

En vez de usar **if** intente con **tryCatch**(vea la sección 2, pagina 52) y observe el comportamiento de la función creada.

Tkinter para abrir ficheros: Intentaremos crear una función para abrir ficheros desde una interfaz.

Ejemplo: 2.1

```
> Abrir <- function(nombre="otro",filtro=".rar"){
    require(tcltk) # paquete tkinter
    lb = paste(
      "{{csv files} {.csv}}",
      "{{spss files} {.sps .sav}}",
      "{{",nombre,"} {",filtro,"}}",
      "{{All files} {*}}", sep=" "
    )
    dir <- tkgetOpenFile(filetypes=lb )
    return(tclvalue(dir))
  }
```

Tkinter para guardar ficheros: Intentaremos crear una funcion para guardar ficheros desde una interfaz.

Ejemplo: 2.2

```
> Guardar <- function(nombre="otro",filtro=".txt"){
   require(tcltk) # paquete tkinter
   lb = paste(
      "{{csv files} {.csv}}",
      "{{spss files} {.sps .sav}}",
      "{{",nombre,"} {",filtro,"}}",
      "{{All files} {*}}", sep=" "
   )
   dir = tkgetSaveFile(filetypes=lb,defaultextension=".csv")
   return(tclvalue(dir))
  }
```

CAPÍTULO 6

Otras Operaciones

1.1 Operador Pipe

El operador de tuberías o pipa “%>%” es muy usado en varios paquetes como es el caso de dplyr, magrittr, ggplot y entre otros. Consiste en concatenar múltiples operaciones hacia un objeto, estas operaciones van separados por “%>%” mientras más este a la izquierda, más superior será:

obj %>% f1 %>% f2 %>% f3 | f3 (f2 (f1 (**obj**)))

Para usar este operado es necesario que instale el siguiente paquete:

```
> install.packages("magrittr")
  ⋮
> library(magrittr)
```

Recuerda conectarse a internet para descargar el paquete, ya después usa solo **library**.

Aquí algunos ejemplos de como se usa:

Ejemplo 1: Desde un objeto se intentará crear una matriz, despúés restaremos 12 a todos los elementos y para finalizar extraemos el elemento máximo.*Nota: solo imprime el cálculo final.*

```
> obj <- c(12,14,23,13,26,34,23,36,49,
         22,14,23)

> obj %>% matrix(nrow=4,ncol=3) %>%
          -12 %>% max
[1] 37
```



"**x%>%f(y)**" es igual a "**x%>%f(x,y)**"

Cuando se trata de funciones como es el caso de la matrix, el operador pipe realiza la operación de esta manera **obj %>%matrix(ncol,nrow)** a esto **matrix(obj,ncol,nrow)**

Ejemplo 2: Usaremos el otro comando **%T>%** para imprimir en consola unicamente la operación que hace cierta función, y se guardará en el objeto "**a**"(**->** **a**)

```
> obj %>% sort %T>% sum  -> a
> a
 [1] 12 13 14 14 22 23 23 23 26 34 36 49
> # Sin el T
> obj %>% sort %T>% sum
[1] 289
```

Ejemplo 3: Si tenemos objetos tipo data.frame, se usará "**%$%**" para reducir comandos como "**tb$a**" o este "**attach(obj)**" que usa como fuente de datos y después solo escribes el nombre de la columna (tedioso si tienes muchos objetos).

```
> obj1 <- c(16,18,26,16,30,44,34,48,56,36,23,37)
# ======  Crear data frame  ======
> tb <- data.frame("a"=obj,"b"=obj1)
# ================================
> tb %$% cor(a,b)
[1] 0.9490482
```

Ejemplo 4: Usar este símbolo "%<>%" para asignar o actualizar valores a un objeto, en caso de **tb** su contenido sera modificado por el resultado de la correlación redondeada a tres cifras.

```
> tb %<>% cor %>% round(digits = 3)
> tb
      a     b
a 1.000 0.949
b 0.949 1.000
```

Este símbolo **%<>%** es muy similar al *ejemplo 2* por el caso de $->a$ que genera el mismo resultado y usted puede comprobarlo usando el símbolo de asignación $<-$ que el mismo R nos ofrece.

Al igual que "magrittr" también hay otra alternativa "**library(pipeR)**" que es muy similar y solo el cambio es "**%>>%**"

2.2 Caracteres

Cada lenguaje de programación tiene en si su propio procesado de caracteres y R no es exclusión. Cuando se habla de caracteres en R nos referimos a objetos que tiene como elementos grupos de textos similares a nombres, fechas, correos y otros casos. El procesado toma en cuenta los espacios vacios

H	o	l	a		M	u	n	d	o
1	2	3	4	5	6	7	8	9	10

Lo comando internos que ofrece R para extraer, reemplazar y recortar caracteres son los que se muestran en la siguiente tabla 6.1

Cuadro 6.1: *Extracción y cambios simples de caracteres*

Función	¿Que hace?
casefold(x, upper)	(x es objeto) Permite cambiar el texto de minúscula(**FALSE**) a mayúscula(**TRUE**) o viceversa.
chartr(old,new,x)	(x es objeto) Cambia un carácter existente(old) por uno nuevo(new).
substr(x,start,stop)	(x es objeto) Extrae grupo de caracteres que están entre start(numérico) y stop(numérico).
strsplit(x,split)	(x es objeto) Recorta caracteres declaradas en split.Genera una lista
regexpr(pattern,text)	(text es objeto) Permite obtener el indice del carácter que coincide con elpatrón(pattern) y la cantidad de repeticiones.También agregue “ignore.case=T” para omitir mayúsculas y minúsculas

Se tomará estos objetos como referencias a los ejemplos:

```
> texto <- "texto en minúscula"

> estado <- c("enojado", "feliz", "enojado", "molesto",
              "enojado", "enojado", "feliz", "mo@esto",
              "feliz",  "molesto", "feliz", "fe@iz")

> correos <- c("mrr@gmail.com","12asa@hotmail.com",
               "5ftds@hotmail.com","H23@gmail.com",
               "qqw2@gmx.com","we222@gmx.com")
```

```
> lb <- paste0("Es malo estar aqui y bien.
¿Usted es el malo?.
Muchos lo dicen aqui y estoy bien ...")
```

Observe los ejemplos:

Ejemplo 1: Cambiar todo el contenido de un objeto a mayúscula y luego a minúscula.

```
> casefold(texto,upper = T)   # igual a -> toupper(texto)
[1] "TEXTO EN MINÚSCULA"
> casefold(texto,upper = F)   # igual a -> tolower(texto)
[1] "texto en minúscula"
```

Ejemplo 2: Verificar si un texto dado como patrón(pattern) se repite en el objeto "correos", si es el caso, hacer un cambio.

```
> grepl(pattern = "@gmx",ignore.case = T,x = correos)
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE

> gsub(pattern = "@gmx",replacement = "@hotmail",x = correos,
     ignore.case = T)
[1] "mrr@gmail.com" "12asa@hotmail.com" "5ftds@hotmail.com"
 ↪  "H23@gmail.com"
[5] "qqw2@hotmail.com"  "we222@hotmail.com"

> chartr(old = "t",new = "T",x = texto) # cambia un carácter
[1] "TexTo en minúscula"
```

Ejemplo 3: Extraer solo una parte del texto dada por el indice y hacer cambios de un solo carácter. También usar **grep** para verificar coincidencias en elementos y luego imprimir dichos valores.

```
> substr(x = texto,start = 10,stop = 18)
[1] "minúscula"
> chartr(old = "m",new = "M",x = texto)
[1] "texto en Minúscula"
#----
> grep(pattern = "@gmail",x = correos)
[1] 1 4
> correos[grep(pattern = "@gmail",x = correos)]
[1] "mrr@gmail.com" "H23@gmail.com"
```

Ejemplo 4: Verificar si hay algún elemento existente dada por patron(pattern) y retornar una lista con elementos entrecortados.

```
> strsplit(x = estado,split = "e", fixed = T)
[[1]]
[1] ""        "nojado"

[[2]]
[1] "f"    "liz"

[[3]]
[1] ""        "nojado"

[[4]]
[1] "mol" "sto"

[[5]]
[1] ""        "nojado"

[[6]]
[1] ""        "nojado"

[[7]]
[1] "f"    "liz"

[[8]]
[1] "mo@" "sto"
```

```
[[9]]
[1] "f"   "liz"

[[10]]
[1] "mol" "sto"

[[11]]
[1] "f"   "liz"

[[12]]
[1] "f"   "@iz"
```

Ejemplo 5: Verificar mediante un patrón si existe “minús” en el objeto *texto* y retornar el índice junto con el tamaño.

```
> regexpr(pattern = "minús",ignore.case = T,text = texto)
[1] 10
attr(,"match.length")
[1] 5
```

En la comunidad de R hay algunas librerías con un gran potencial para el análisis de texto (NLP) “Procesamiento de Lenguaje Natural”. Para continuar por favor instale las siguientes librerías

```
> install.packages("stringr")
> install.packages("stringi")
> install.packages("tokenizers")
> install.packages("tidyverse",dependencies = T)
> install.packages("cleanNLP")

    ⋮
> library(tidyverse)
> library(stringr)
> library(stringi)
> library(tokenizers)
> library(cleanNLP)
```

Para mas información de *stringr* visite (**https://stringr.tidyverse.org/**) o la web github (**https://github.com/tidyverse/stringr**)
Si tienes dificultades en tidyverse, instala
`install.packages("rlang", type = "source")`

Ejemplo 6: Usar librería **stringr**

```
# Contar cantidad de caracteres
> str_count(string = correos)
[1] 13 17 17 13 12 13

#  Contar solo los elementos que tienen a hotmail
> str_count(string = correos,pattern = "hotmail")
[1] 0 1 1 0 0 0

#  Mostrar solo los elementos que terminan en hot
> str_subset(string = correos,pattern = "hot")
[1] "12asa@hotmail.com" "5ftds@hotmail.com"

# Si quieres verlo en html tipo web
> install.packages("htmlwidgets")
```

```
> str_view(string = correos,pattern = "hotmail")
```

Ejemplo 7: Usar librería **stringi**

```
# Verificar si si existe una letra
> correos %stri<% "b"
[1] FALSE  TRUE  TRUE FALSE FALSE FALSE

#  Mostrar estadísticas generales
> stri_stats_general(str = texto)
      Lines LinesNEmpty       Chars CharsNWhite
          1           1          18          16

#  Mostrar estadísticas especificas
> stri_stats_latex(str = texto)
    CharsWord CharsCmdEnvir    CharsWhite          Words
           16             0             2              3
         Cmds        Envirs
            0             0
```

Ejemplo 8: Usar librería **tokenizers** para ver palabras

```
# Verificar si si existe una letra
> tokenize_words(x = lb)
[[1]]
 [1] "bien"           "querido"        "fagin"          "se"
 [5] "puede"          "saber"          "que"            "haces"
 [9] "ha"             "esta"           "hora"           "de"
[13] "la"             "noche"          "visitando"      "este"
[17] "vecindario"     "y"              "encima"         "has"
[21] "logrado"        "traer"          "el"             "gran"
[25] "botin"          "que"            "supuestamente"  "era"
[29] "para"           "las"            "cuatrocientas"  "horas"
[33] "si"             "deseas"         "un"             "cambio"
```

```
[37] "espera"          "esta"            "noche"
```

Ejemplo 9: Verificar cuantas veces se repite las palabras. Se usará *tibble* en vez de *data.frame*

```
# ==== library("tidyverse") ====
> a <- tokenize_words(x = lb)
> b <- table(a[[1]])
> tibble(letras = names(b),repetido = as.numeric(b))
# A tibble: 12 x 2
   letras repetido
   <chr>     <dbl>
 1 aqui          2
 2 bien          2
 3 dicen         1
 4 el            1
 5 es            2
 6 estar         1
 7 estoy         1
 8 lo            1
 9 malo          2
10 muchos        1
11 usted         1
12 y             2
```

Ejemplo 10: Contar palabras, caracteres y sentencias

```
# ====     ====
> count_words(x = lb)
[1] 17
> count_sentences(x = lb)
[1] 3
> count_characters(x = lb)
[1] 84
```

Ejemplo 11: Usar **cleanNLP** para un análisis en profundidad, *Atención: Se necesita instalar Python y después usar en consola* **pip install cleannlp**

```
# ====     ====
> cnlp_init_stringi(locale="es_ES")
> cnlp_init_udpipe(model_name="spanish")
> cnlp_init_spacy(model_name="es")
Error: Python module 'cleannlp' not found. Install with:
  pip install cleannlp
> cnlp_init_corenlp(lang="es")
Error: Python module 'cleannlp' not found. Install with:
  pip install cleannlp
> #--- ejemplo
> cnlp_annotate(input = lb)

$token
# A tibble: 22 x 11
 sid tid token token_with_ws lemma upos xpos feats tid_source relation
<int> <chr> <chr> <chr>       <chr> <chr> <chr> <chr> <chr>     <chr>
 1      1    Es     "Es "        ser   AUX    NA    Mood... 2         cop
 1      2    malo   "malo "      malo  ADJ    NA    Gend... 0         root
 1      3    estar  "estar "     estar AUX    NA    Verb... 2         csubj
 1      4    aqui   "aqui "      aqui  ADV    NA    NA      3         advmod
 1      5    y      "y "         y     CCONJ  NA    NA      6         cc
 1      6    bien   "bien"       bien  ADV    NA    NA      4         conj
 1      7    .      ".\n"        .     PUNCT  NA    NA      2         punct
 2      1    ¿      "¿"          ¿     PUNCT  NA    NA      5         punct
 2      2    Usted  "Usted "     tú    PRON   NA    Case... 5         nsubj
 2      3    es     "es "        ser   AUX    NA    Mood... 5         cop
# ... with 12 more rows

$document
  doc_id
1      1
```

3.3 Gráficos

Hay muchas librerías en la comunidad del lenguaje R ofreciendo generar gráficos de distintos tipos de diseño. Es importante tener todas las librerías y el programa R actualizados. Para esta sección se necesitará estas librerías

```
> install.packages("ggrepel") # ggplot2 comp.
> install.packages("wordcloud2")
> install.packages("webshot")
> install.packages("ggplot2")
> install.packages("plotly")
> webshot::install_phantomjs()
```

1.3.1 Wordcloud

Genera una nube de palabras de forma aleatoria, también puedes agregar una imagen como contorno.

Wordcloud:

```
> library(webshot) #sirve para generar html
> library(wordcloud2)
> library(tokenizers)
```

```
lb = paste0("hombre de los garabatos arremetió contra los
          gatos,hombre de los gatos arremetió garabatos,
          los garabatos del hombre arremetió contra los gatos,
          los gatos son garabatos")
#
a <- tokenize_words(x = lb)
b <- table(a[[1]])
tb <- tibble(letras = names(b),repetido = as.numeric(b))
#ar <-
wordcloud2(data = tb,size = 1.8 )
```



Cuadro 6.2: Formas en wordcloud

Forma	Produce
circle	Un circulo
cardioid	Un gráfico cardioide
diamond	Un diamante
triangle-forward	Un triangulo hacia adentro
triangle	Un triángulo
pentagon	Un pentágono
star	Una estrella

Nota: Para obtener la forma correcta es necesario **tener hojas de puro palabras** ya que depende de la cantidad de palabras.

```
#
> wordcloud2(data = tb,shape="circled")
#
```



"Exportar a html y pdf"

Es muy importante generar un pdf o cualquier otro formato de imagen y se debe a que cuando usted esta creando un reporte en Sweave pdf, trabaje correctamente dado a que *wordcloud* solo genera un formato web. Es aquí donde usa la librería *webshot* para tomar captura.

```
# librerias
> library(webshot)
> library(htmlwidgets)
        ⋮
#---------------------
> img <- wordcloud2(data = tb,shape="circled")

#
> saveWidget(img, "temp.html",selfcontained = F)
> webshot("temp.html",file="temp.pdf", cliprect="viewport")
#
```

Información y ayuda:

Puede usted encontrar más información en la siguiente web:
https://www.r-graph-gallery.com/196-the-wordcloud2-library
Si usted tiene instalado inkscape y desea una mejor calidad al exportar, use esta web:
https://www.jasondavies.com/wordcloud/

2.3.2 Plotly

Una librería muy interesante que esta disponible en lenguajes de programación como: Python, Javascript, Matlab, Perl, Julia, Arduino y R. Como nos encontramos en R entonces se necesita incluir esta librería *plotly* para las gráficas y dplyr es alternativa para crear el cuadro de datos (tibble) que ante se usaba `data.frame`



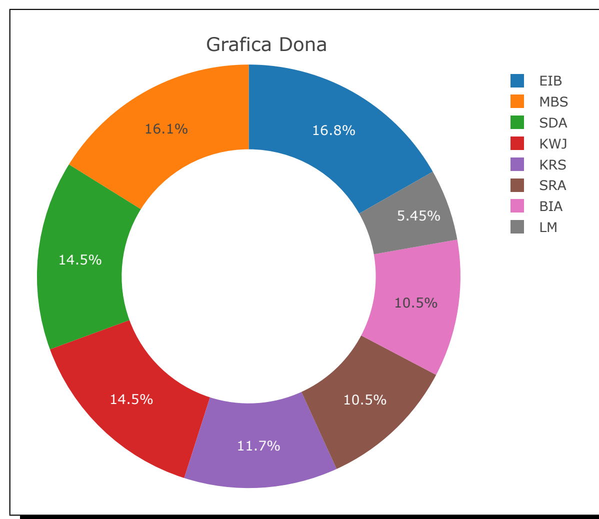
```
> library(plotly)
> library(dplyr) # para tibble
```

La librería genera un fichero web(html) que solo se puede abrir con el navegador web(si es que no cuenta con Rstudio).

Observe los siguientes ejemplos:

Gráfico de donas

```
> tab <- tibble(nombre= c("MBS","KRS","LM","SRA",
                          "EIB","BIA","SDA","KWJ"),
                valor = c(77, 56, 26, 50, 80, 50, 69, 69))
> tab
# A tibble: 8 x 2
  nombre valor
  <chr>  <dbl>
1 MBS       77
2 KRS       56
3 LM        26
4 SRA       50
5 EIB       80
6 BIA       50
7 SDA       69
8 KWJ       69
#================
                                    1
> fg <- tab %>% plot_ly(labels=~nombre,values = ~valor) %>%
          add_pie(hole = 0.6)
#-----            2
> fg %>% layout(title = "Grafica Dona",showlegend = T)
> fg # ver grafico
```



Se toma los elementos de *tab* para luego ser procesados en plot_ly, recuerda que todo esto es almacenado en un nuevo objeto llamado ***fg***.

En **1** el argumento **labels** hace referencia a las etiquetas que en la data(tb) se encuentra en la columna llamada "nombre" pero he aquí un detalle, presta atención al símbolo empleado (~nombre = **tab$**nombre) ambos generan el mismo resultado. Al igual que labels, **values** es en efecto el mismo caso solo que en vez de etiqueta seria los datos numericos.

Luego se encuentra la función **add_pie(hole = 0.6)** que significa el grosor de la dona, mientras mas se aproxime a la unidad (1.0) mas fino será el relleno.

En **2** configuramos el diseño como es el caso del titulo,nombres al eje x e y, la leyenda y entre otros.

Gráfico Pie(torta)

```
> fg <- tab %>% plot_ly(labels=~nombre,values = ~valor,
                type= "pie",textinfo= 'label+percent',
                insidetextorientation='radial',
                domain= list(row=0,column=1)) %>%
          layout(title="Grafico Pie",showlegend=T)
```

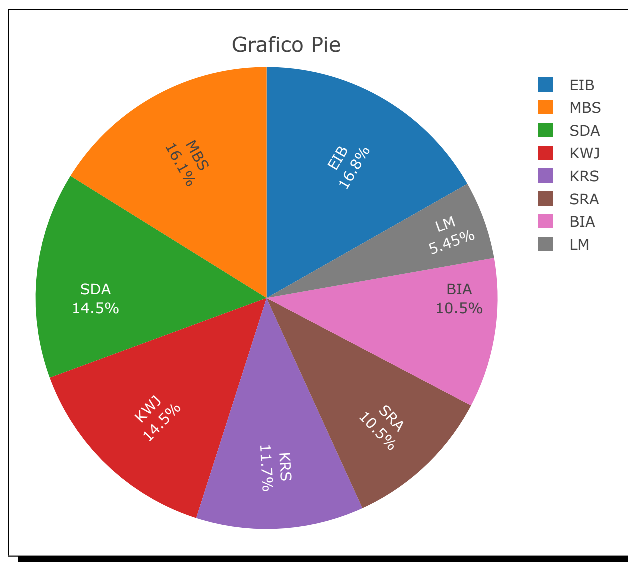


Gráfico de Dispersión

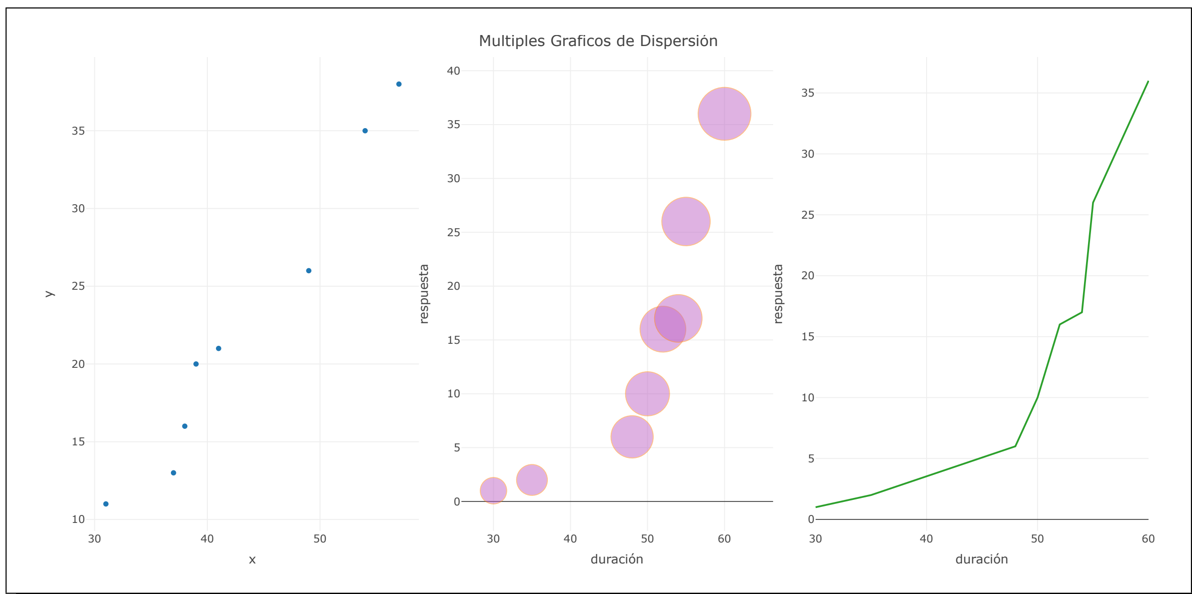
```
> tb <- tibble(x = c(31, 37, 38, 39, 41, 49, 54, 57),
               y = c(11, 13, 16, 20, 21, 26, 35, 38),
             duración = c(30, 35, 48, 50, 52, 54, 55, 60),
             respuesta = c(1,  2,  6, 10, 16, 17, 26, 36))

> fg1 <- tb %>% plot_ly(x = ~x,y = ~y,type = 'scatter',
                   mode='markers')

> fg2 <- tb %>% plot_ly(x = ~duración,y = ~respuesta,
                  type = 'scatter', mode='markers',
                  marker = list(size=~duración,
                  opacity=0.5, color ='#bf66c6'))

> fg3 <- tb %>% plot_ly(x = ~duración,y = ~respuesta,
                  type = 'scatter', mode='lines')

> subplot(fg1,fg2,fg3, titleX = T, titleY = T,nrows=1) %>%
      hide_legend() %>%
      layout(title="Multiples Graficos de Dispersión")
```

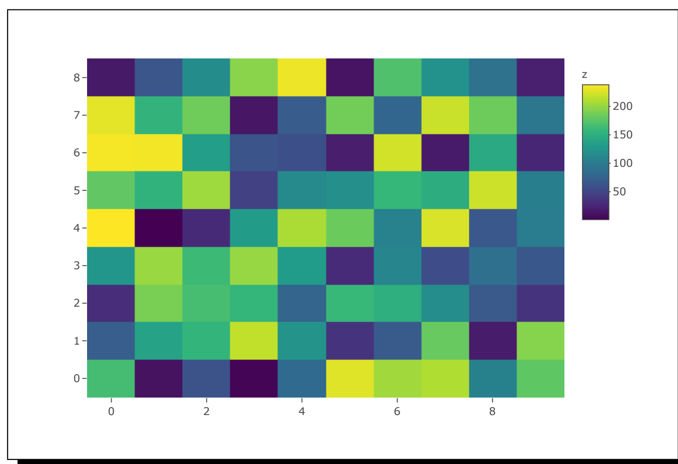


Cuadro 6.3: *Colores compatibles con plotly*

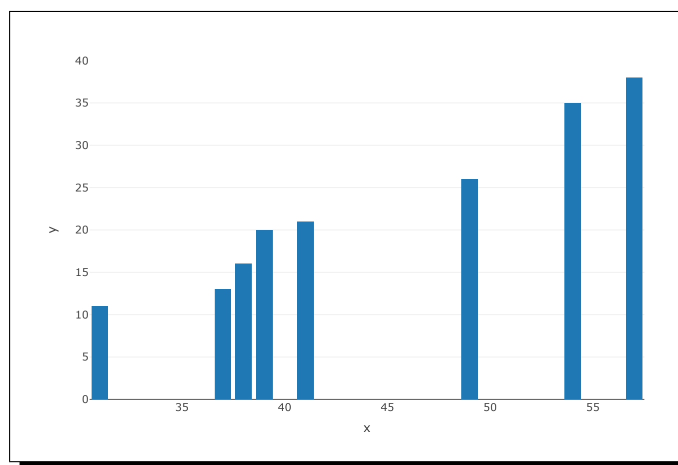
color	hex	`color = "#FF0000"`
	rgb	`color = "rgb(98,197,59)"`
	rgba	`color = "rgba(98,197,59,0.5)"`
	hsl	`color = "hsl(335, 52%, 37%)"`
	hsv	`color = "hsv(117,89%,100%)"`

Importante: subplot( ) parece ser que no funciona con gráficos tipo **pie** pero si con el resto.

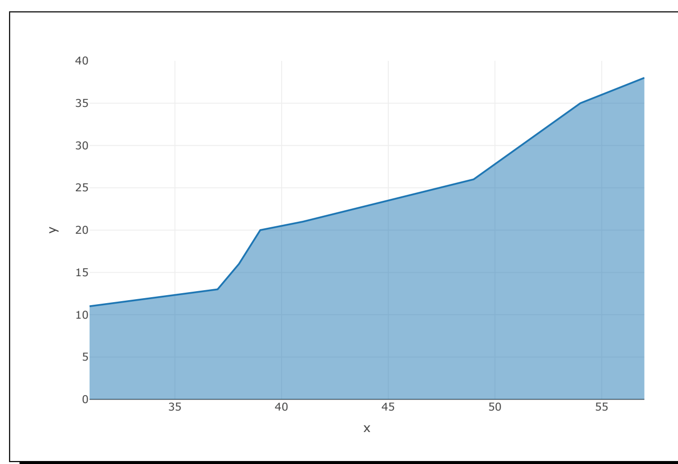
```
#-----------------------------
> a <- sample(1:240,size = 90,replace = F)
> tabla <- tibble(z = matrix(data = a,ncol = 10) )
> plot_ly(data = tabla,z=~z, type = 'heatmap')
# mapa de calor
#-----------------------------
```



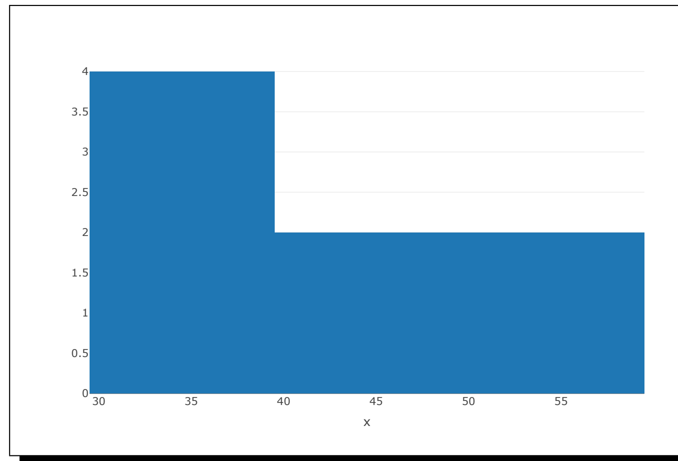

```
#-----------------------------
> plot_ly(data = tb, x=~x,y=~y, type = "bar", mode ="markers"
↪ )
# barras
#-----------------------------
```



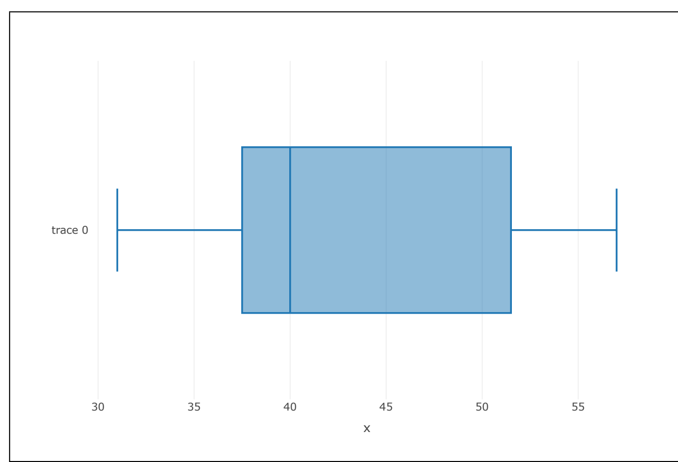

```
#-----------------------------
> plot_ly(data = tb, x=~x,y=~y,type = "scatter",
        mode="lines",fill="tozeroy")
# dispersion con relleno
#-----------------------------
```



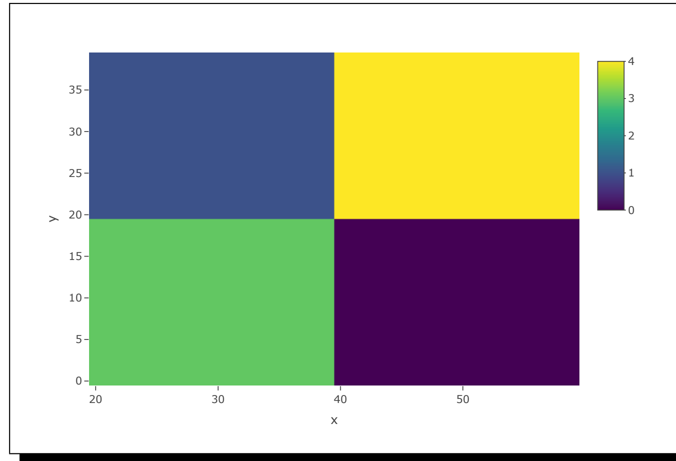

```
#-----------------------------
> plot_ly(data = tb,x=~x, type = "histogram")
# histograma
#-----------------------------
```




```
#----------------------------
> plot_ly(data = tb,x=~x, type = "box")
# grafico boxplot
#----------------------------
```




```
#----------------------------
> plot_ly(data = tb, x=~x, y=~y, type = "histogram2d")
# grafico histograma en 2d
#----------------------------
```





"Mas información"

En su pagina web **https://plotly.com/r/** encontrará todos los gráficos disponibles y si no es suficiente entonces descargue el cheat sheet que esta disponible en **https://images.plot.ly/plotly-documentation/images/r_cheat_sheet.pdf**

3.3.3 GGplot

Es una de las librerías muy usadas y esto es porque permite personalizar cada detalle en la gráfica, lo de bueno es que trabaja modo sin conexión permitiendo exportar la gráfica en diferentes formatos como: "eps, ps, tex (pictex), pdf, jpeg, tiff, png, bmp, svg y wmf".



```
> library(ggplot2)
> library(ggrepel) #geom_label_repel()
```

Importante: El signo **+** sirve para unir funciones, esto es muy similar al operador pipe.

Cuadro 6.4: *Gráficos para una Variable*

Código	**Produce**
p + geom_density( )	gráfico de densidad
p + geom_dotplot( )	gráfico de puntos
p + geom_histogram(binwidth = n)	histograma
p + geom_bar()	gráfico de barras
p + geom_density()	gráfico de densidad

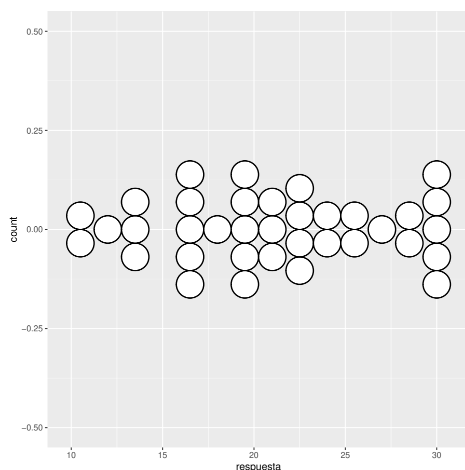
Nota: ***n*** es un numero cualquiera

data **tb**

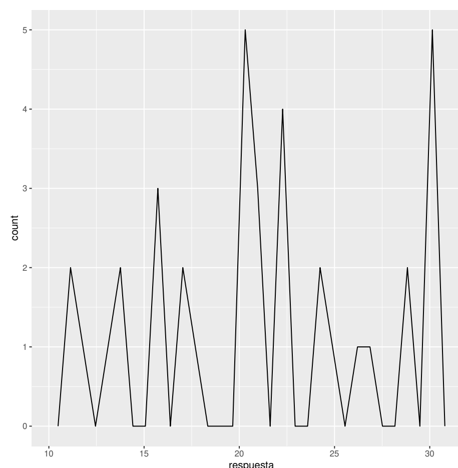
```
> tb <- tibble(duracion = sort(sample(10:30,
                                  size=36,replace=T)),
               respuesta = sort(sample(10:30,
                                  size=36,replace=T)) )
#------
> p <- ggplot(data = tb,aes(x = respuesta))
```

Ejemplos usando una sola variable

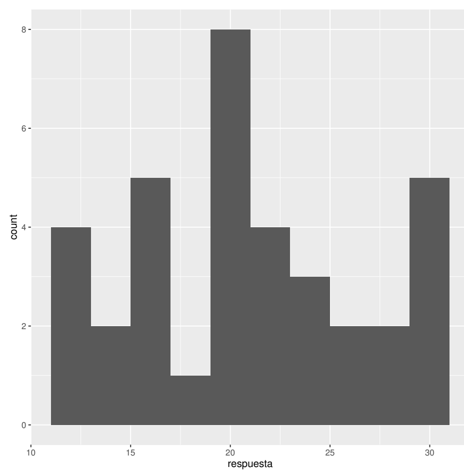
```
> p+geom_dotplot(binwidth=2,
      method="histodot",
      stackdir="center",
      fill="white",stroke=2)
```



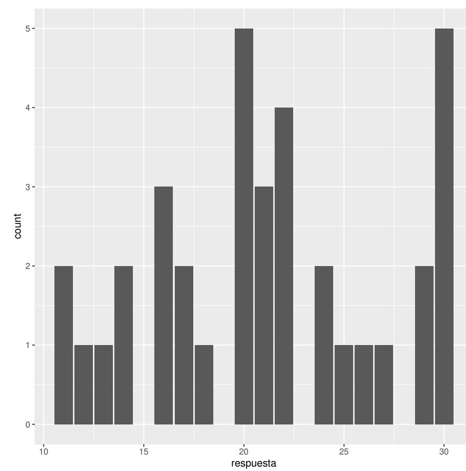

```
> p + geom_freqpoly()
```



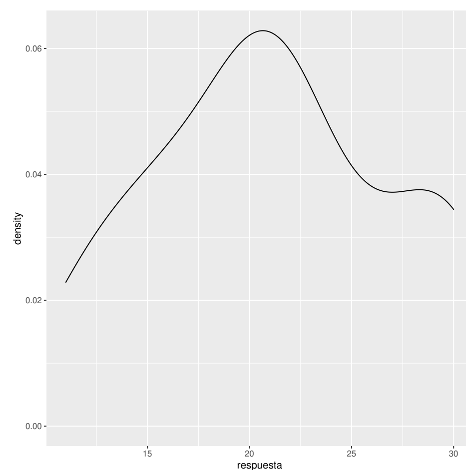
```
>
↪ p+geom_histogram(binwidth=2)
```




```
> p + geom_bar()
```




```
> p + geom_density()
```



El siguiente gráfico estadístico es el cuantil junto con la pendiente (stat_qq_line)

```
> p <- ggplot(data = tb,aes(sample = respuesta))
> p + stat_qq() + stat_qq_line()
```

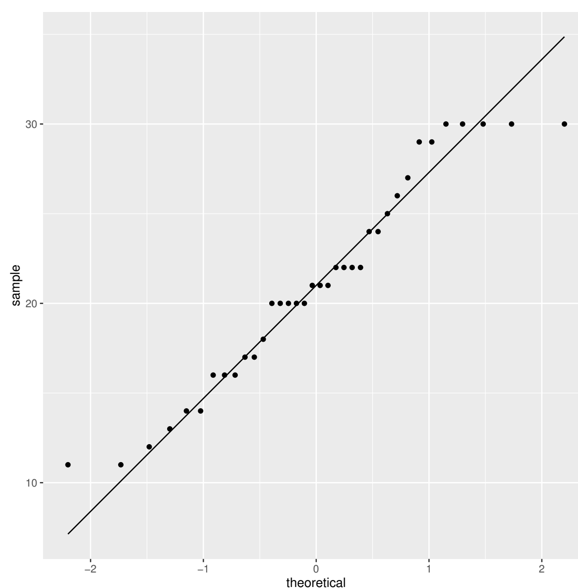


Gráfico Pie

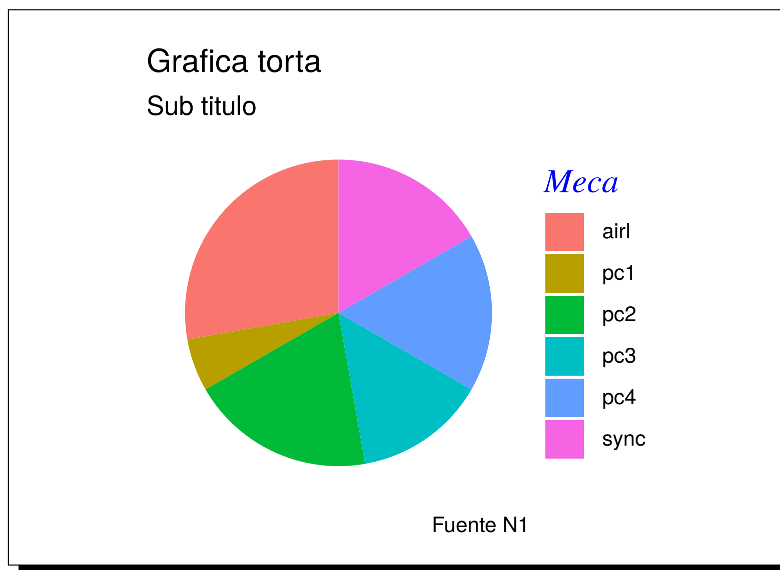
También esta la gráfica de torta pero hay todo un proceso para lograr el diseño, al igual que el anterior se intentará crear un objeto temporal que nos servirá para generar valores aleatorios y después incluirlo en **tb**

Agregar nueva columna a data **tb**

```
> med <- sample(c("airl","pc1","pc2","pc3","pc4","sync"),
                 size=36, replace=T)
> tb$m <- med
> tb
# A tibble: 36 x 3
   duracion   respuesta    m
      <int>       <int>  <chr>
 1       10          11    pc4
 2       10          11    pc2
 3       10          12    sync
 4       11          13    pc3
# ... with 32 more rows
```

Por tratarse de gráficos de una sola variable, usaremos solo la columna "**tb$m**". Se necesitará de la función **geom_bar( )** con el **coord_polar( )** y un tema que permite eliminar ciertas partes **theme_void( )**

```
> t <- ggplot(data = tab,aes(x = 1,fill= m))
> t + geom_bar(width = 1) + coord_polar(theta="y",start=0) +
      theme_void() + labs(
        title = "Grafica torta",
        subtitle = "Sub titulo",caption = "Fuente N1",
        fill ="Meca") +
      theme(legend.title = element_text(
        face = "italic",
        family = "Times",
        colour = "blue",
        size = 14
      )# ,  1
  )
```



Recuerda que el mismo procedimiento de *legend.title* se aplica a continuación del 1 para los siguiente:

plot.title( ) −− > titulo
plot.subtitle( ) −− > subtitulo
plot.caption( ) −− > encabezado

La ubicación de la leyenda es definida por:
legend.position = −− > `"top"`, `"right"`, `"left"`, `"bottom"` SOLO ELEGIR

UNO.

La siguiente función permite encerrar el gráfico en un cuadro
`plot.background = element_rect(colour = "orange",size = 2)`

Previa de como se debe usar

```
> t + geom_bar() + coord_polar() + theme_void() + labs() +
      theme(
        legend.title = element_text(),
        plot.title = element_text(),
        plot.subtitle = element_text(),
        plot.caption = element_text(),
        legend.position = "top",
        plot.background = element_rect()
      )
```

Usted debe intentar dar forma a sus graficos probando distintas funciones ya mostradas anteriormente. Ahora pasemos a los gráficos que necesitan de 2 variables, vea la siguiente tabla 6.5

Cuadro 6.5: *Gráficos para dos Variables*

Código	**Produce**
p + geom_area( )	Gráfico de área con relleno
p + geom_bar( )	Gráfico de barras
p + geom_boxplot( )	Gráfico de cajas
p + geom_rug( )	Gráfico de lineas en ejes, usalo como complemento
p + geom_jitter( )	Similar a geom_point( ), con la diferencia que agrega puntos aleatorios para datos dispersos
Distribución Bivariado	
p + geom_bin2d( )	Mapa de calor con cuadros 2d
p + geom_density2d( )	Gráfico de densidad en dos dimensiones
p + geom_hex( )	Mapa de calor con cuadros hexagonales similar a geom_bin2d( )
p + geom_area( )	Gráfico de relleno con lineas tipo cintas
p + geom_line( )	Gráfico de lineas

Nota: en caso de ser gráfico de puntos, puedes usar “`alpha = 0.6`”(*va de 0.0 a 1.0*)

Pasemos a algunos ejemplos:

2 variables numéricos y 1 no numérico

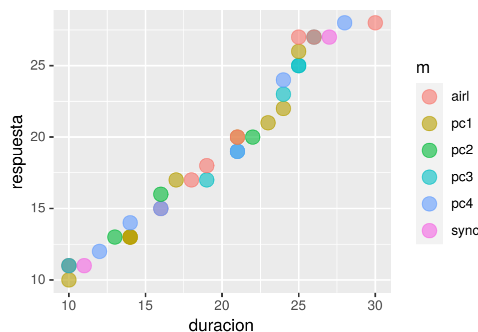
```
> p <- ggplot(data=tb,aes(x=duracion,y=respuesta,color = m ))
```

Agregando en *aes*(color=objeto$columna), crear colores aleatorios de acuerdo a la columna, es de vital importancia que la columna seleccionada tenga elementos repetitivos.

Ejemplos usando dos variable

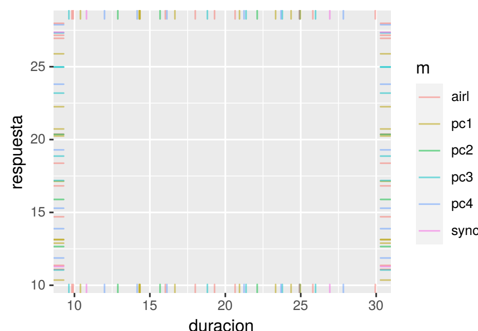
```
> p + geom_point(size=4,
    alpha=0.6)
```

el argumento **size** sirve para reducir o aumentar el tamaño de los puntos (*no usar valores negativos*)

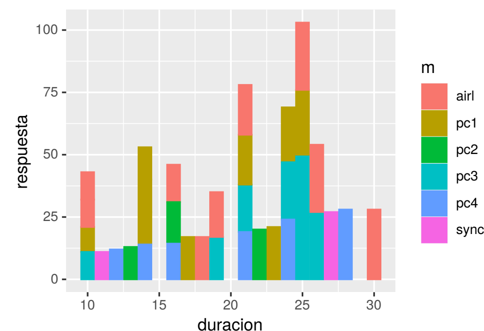



```
> p + geom_rug(alpha=0.5,
    sides="trbl",
    position="jitter")
```

alpha es la transparencia del color(valores 0.0 hasta 1.00).

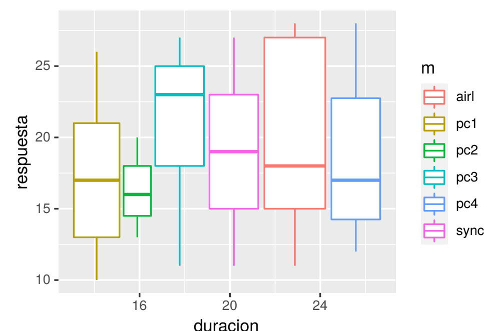



```
> p + geom_bar(
    stat="identity",
    aes(fill=m))
```

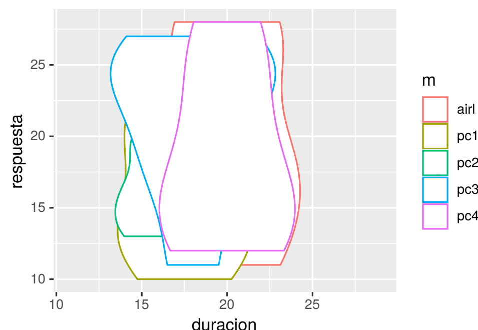



```
> p + geom_boxplot()
```

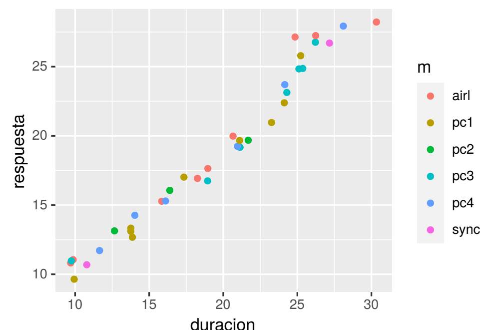
el aes(fill=m) sive para ajustar y rellenar de color las barras de acuerdo a la columna m.



```
> p + geom_violin()
```

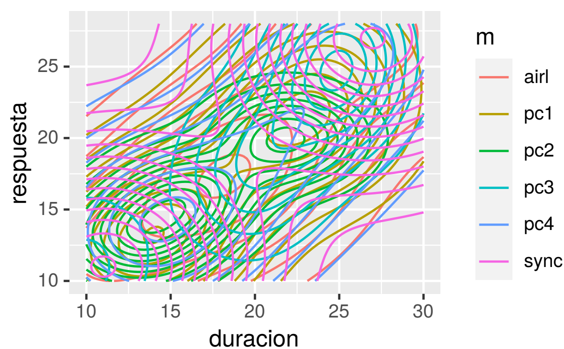


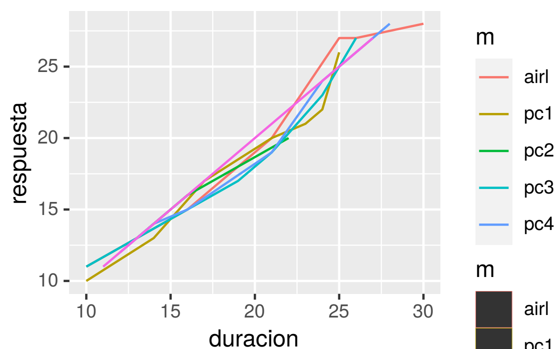

```
> p + geom_jitter()
```

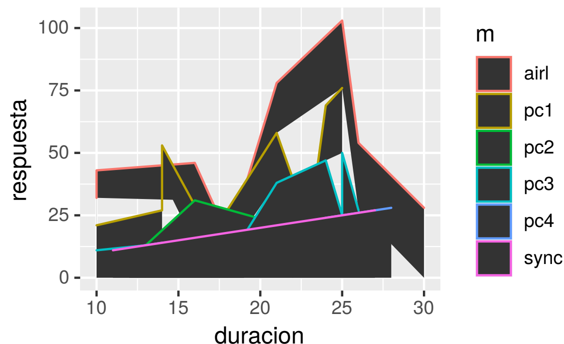


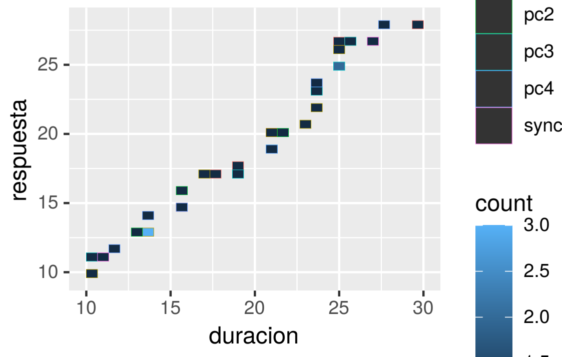
Hay una función llamada *grid.arrange(...,ncol=n)* que permite unir múltiples gráficas en una sola, por ejemplo:

```
> q4 <- p + geom_bin2d()
> q1 <- p + geom_density2d()
> q3 <- p + geom_area()
> q2 <- p + geom_line()
> gridExtra::grid.arrange(q1,q2,q3,q4)#usar ncol= 2
```



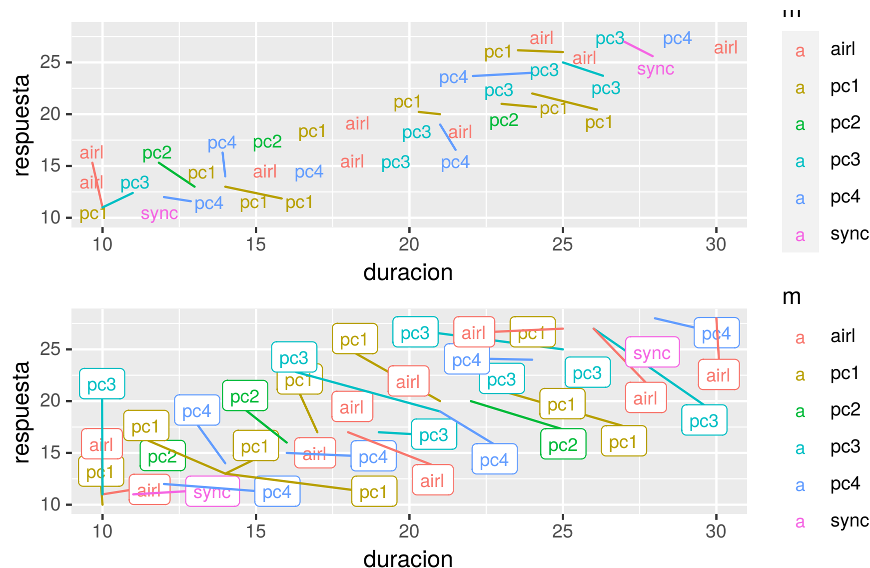






Aquí usaremos la librería **library(ggrepel)** que ofrece dos funciones: **geom_text_repel()** muestra solo texto y **geom_label_repel()** muestra el texto dentro de un rectángulo.

```
#library(ggrepel) #
> l1 <- p + geom_text_repel(aes(label=m), size= 3)
> l2 <- p + geom_label_repel(aes(label=m), size= 3)
> gridExtra::grid.arrange(l1,l2)
```

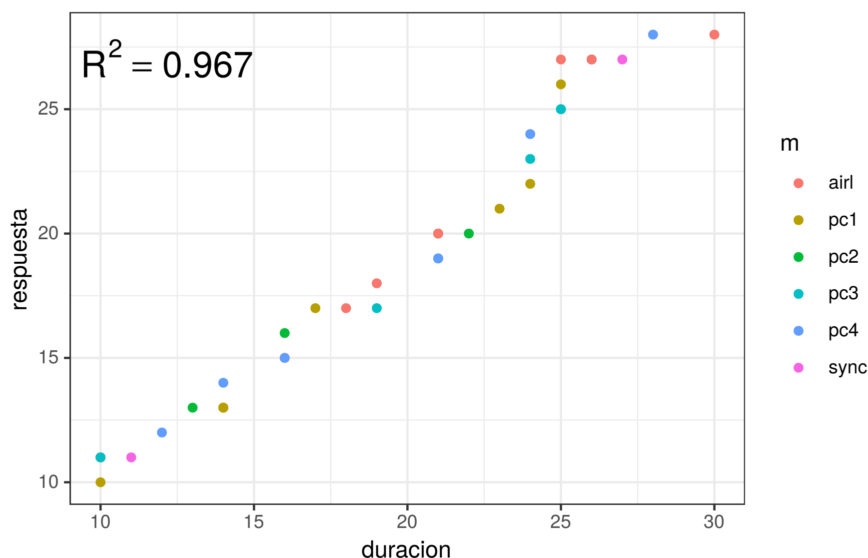


Annotate

Los textos matemáticos también se pueden escribir en ggplot pero tiene su propia sintaxis de escritura, si desea más información visite "**https://astrostatistics.psu.edu/su07/R/html/grDevices/html/plotmath.html**"

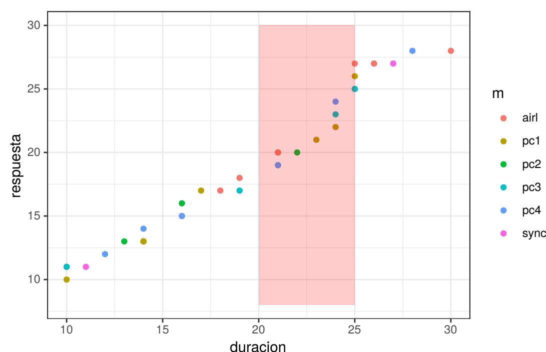
```
#---- formula para calcular R^2
> r = round(summary(lm(data = tb,
    formula = duracion~respuesta))$r.squared,3)
#--------------------------------
> p + geom_point() + annotate("text",x=12,y=27,parse=T,
    label=paste("~R^2 ==",r ),size=6) + theme_bw()
```

Se ha usado ***annotate(parse=T)*** es importante que *parse=T* solo de esa forma imprime texto matemático, la facilidad de esta función es que permite ubicar el texto en los puntos "*x*" e "*y*". También hay otras opciones usando `angle=90,colour="red",family="Courier", fontface="bold"`



Annotate permite usar la opción "*rect*" que permite crear un rectángulo a partir de las dimensiones dadas

```
> p + geom_point() +
  annotate("rect",
    xmin=20, xmax=25,
    ymin=8,ymax=30,
    alpha=0.2 ,fill="red") +
  theme_bw()
```



la otra opción que crea una flecha en el gráfico, es "*segment*" y muy importante cuando se trata de gráficos estadísticos

```
> p + geom_point() +
  annotate("segment",
    x=16, xend=19,
    y=30,yend=26,
    alpha=0.3 ,fill="red",arrow=arrow(),size=2)
```

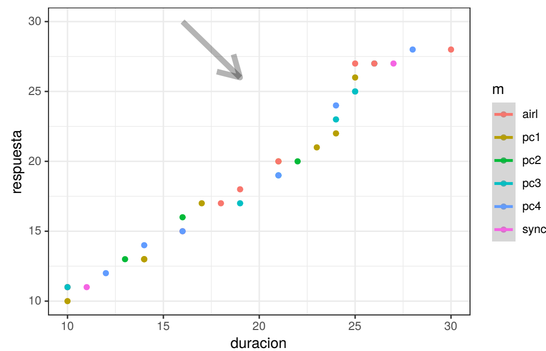


gráfico torta

Continuando con gráficos tipo torta, en este apartado se intentará mejorar el diseño usando dos variables(columnas) por motivos de estética.

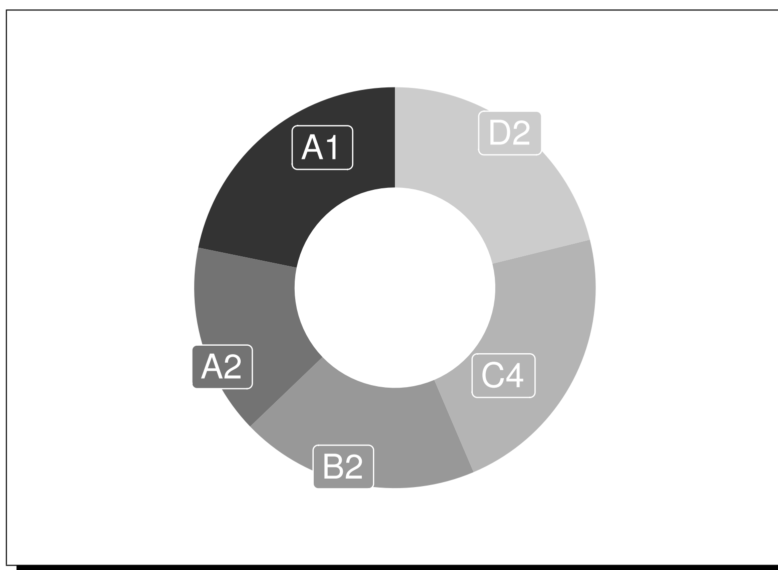
data para gráfico pie usando 2 variables

```
> tab <- tibble(valor=sample(20:40,size = 5,replace = T),
                text= c("A1","A2","B2","C4","D2") )
#---- prop = "proporcion"
> tab <- tab %>% arrange(desc(text)) %>%
                mutate(prop = valor/ sum(tab$valor)*100)
#----- posicion de "y" para label
> tab <- tab %>% mutate(ypos = cumsum(prop)- 0.5*prop)
> tab
# A tibble: 5 x 4
  valor text   prop   ypos
  <int> <chr> <dbl>  <dbl>
1   33     D2  21.2   9.52
2   35     C4  22.4  31.2
```

```
3   30    B2   19.2  52.2
4   24    A2   15.4  69.7
5   34    A1   21.8  88.0
```

```
> torta <- ggplot(data = tab,aes(x = 2, y=prop, fill= text))

> torta + geom_bar(stat="identity",width=1, show.legend=F) +
    coord_polar(theta = "y", start = 0) + theme_void() +
    geom_label_repel(aes(y = ypos,label=text),color="white",
            size= 5,show.legend = F
    ) + scale_fill_grey() + xlim(0.5, 2.5)
```



Usando otra forma y con colores definidos manualmente por **scale_fill_manual(values = objeto)**

```
> col <- c("#31f48e","#07aa7b","#0bbeff","#0086ff","#01437d")

> ggplot(tab, aes (x="", y = valor, fill = factor(text))) +
    geom_bar(width = 1, stat = "identity") +
    geom_text(aes(
        label = paste(round(valor/sum(valor) * 100,1),"%")),
```

```
        position = position_stack(vjust = 0.5),
        color="white",fontface="bold") +
    theme_classic() +
    theme(plot.title = element_text(hjust=0.5),
        axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank()) +
    labs(fill = "Categoria",
        x = NULL,
        y = NULL,
        title = "Grafico Circular") +
    coord_polar("y")+ scale_fill_manual(values = col)
```

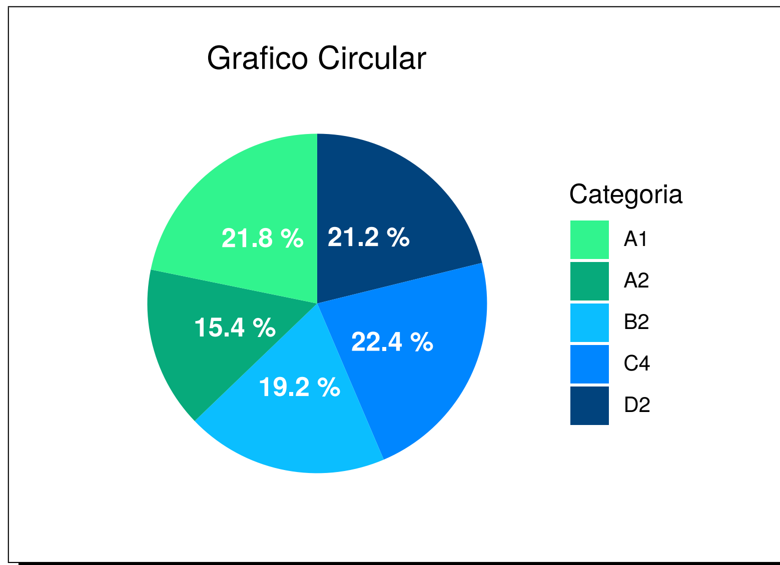


gráfico de barras para escala likert

Además de crear estos gráficos sencillos, intentaremos ir un poco mas lejos y para ello se intentará crear un gráfico tipo likert con tres dimensiones, nos apoyaremos por el gráfico de barras:



Importante

Es necesario descargar estos dos ficheros:
stat_fill_labels.R

inv_cumsum.R
desde su pagina github del autor **https://github.com/larmarange/JLutils/tree/master/R** o también en esta parte **https://github.com/Moriand/Rlabels**

Una vez descargado use esto en su consola de R

```
> source(file = file.choose()) #stat_fill_labels.R
> source(file = file.choose()) #inv_cumsum.R
```

Primero de lo primero, se necesita la data y lo obtendremos con una función que genera valores aleatorios:

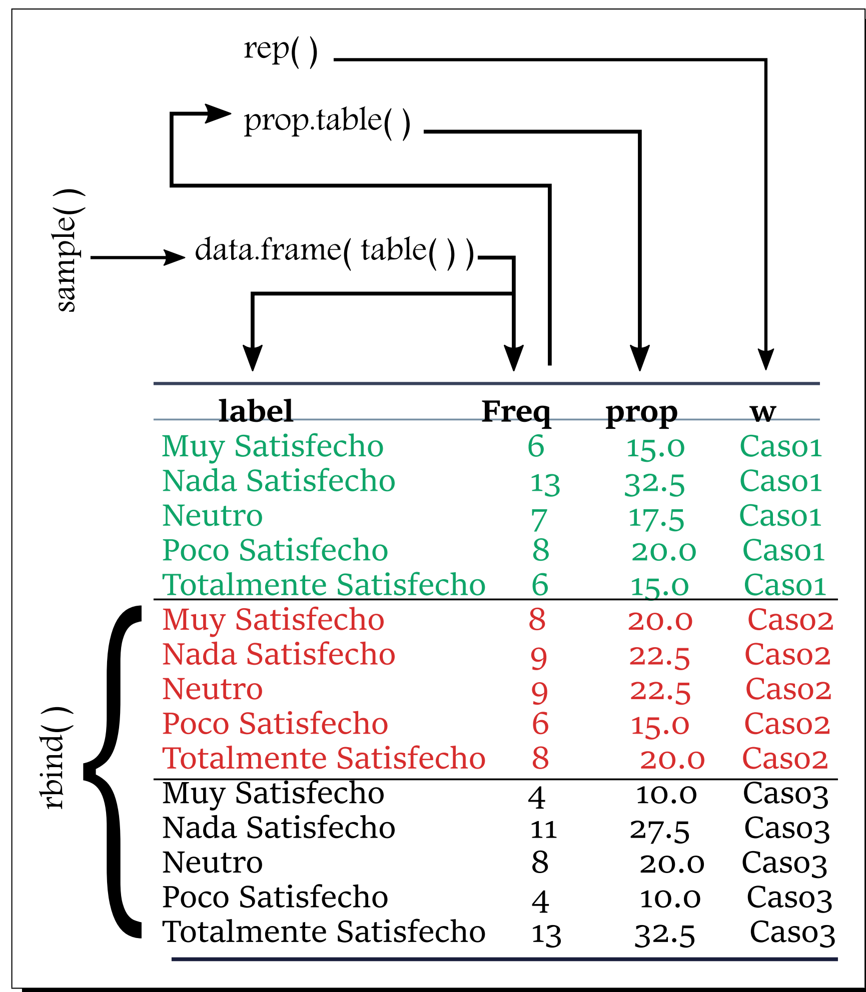
función para generar valores aleatorios

```
> funcion <- function(caso,lb,n){
    tmp <- sample(x = lb,size = n,replace = TRUE) %>%
            table() %>% data.frame()
    tmp$p <- tmp$Freq %>% prop.table() %>%
            round(digits = 4)*100
    tmp$w <- rep(caso,length(tmp$Freq))
    names(tmp) <- c("label","Freq","prop","w")
    return(tmp)
  }
```

Intentaremos crear un objeto que contenga las etiquetas likert (*lab*) y un objeto llamado casos que se encargara de almacenar la data

data casos

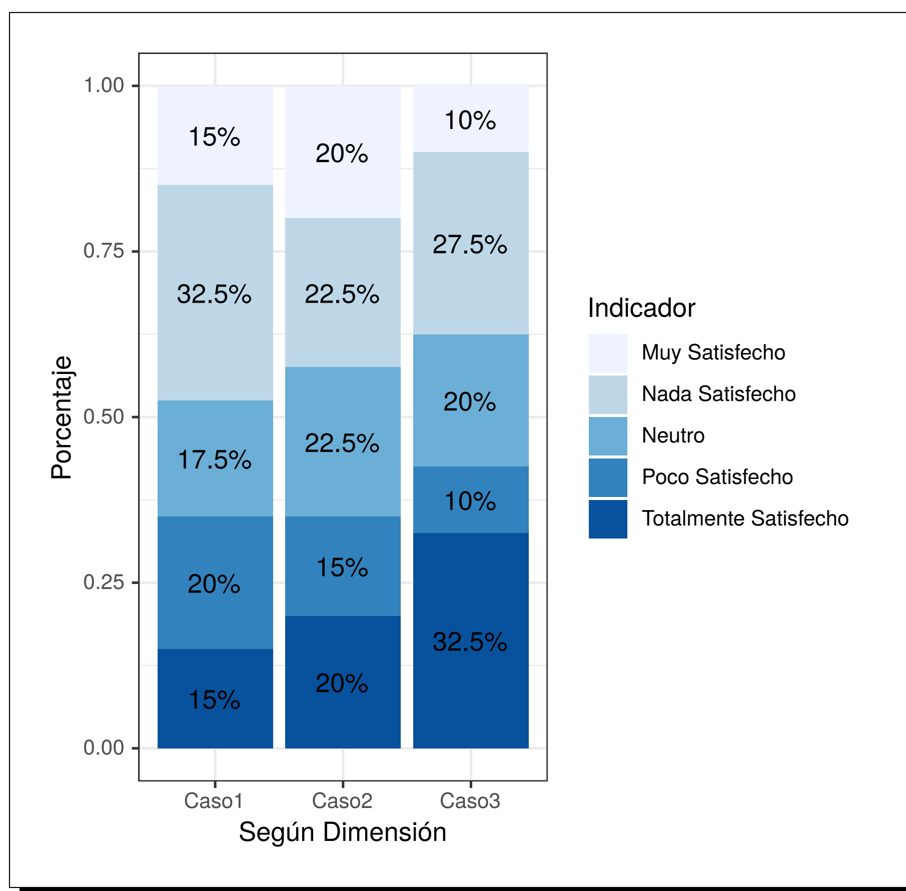
```
> lab <- c("Totalmente Satisfecho","Muy Satisfecho",
        "Neutro","Poco Satisfecho","Nada Satisfecho")
> casos <- funcion(caso = "Caso1",lb = lab,n = 40)
> casos <- rbind(casos,funcion(caso = "Caso2",
                                    lb = lab,n = 40))
> casos <- rbind(casos,funcion(caso = "Caso3",
                                    lb = lab,n = 40))
```



Usamos la data *casos* que es un data.frame, en la librería *ggplot* de la siguiente manera

grafico

```
# stat_fill_labels.R
# inv_cumsum.R
> s <- ggplot(data=casos,aes(x=w,fill=label,weight=prop))
> s + geom_bar(position = "fill") + scale_fill_brewer() +
    theme_bw() +
    labs(
        fill="Indicador",face="bold",size=14,
        y="Porcentaje",x="Según Dimensión") +
    stat_fill_labels()
```



Puede agregar rectángulos a cada etiqueta que contiene el porcentaje cambiando
`stat_fill_labels()` por este otro
`geom_label(stat = "fill_labels")`

temas

La librería ofrece varios tipos de temas que cada uno de ellos tiene distintos estilos.

Cuadro 6.6: *Lista de temas para ggplot*

temas	temas
theme_bw( )	*theme_grey*( )
theme_classic( )	*theme_light*( )
theme_dark( )	*theme_linedraw*( )
theme_void( )	*theme_minimal*( )
theme_gray( )	*theme_test*( )
argumentos	
`base_size = 14`	tamaño del texto
`base_family = "times"`	fuente tipográfica para el texto
`base_line_size = 9`	grosor de linea para los ejes x e y
`base_rect_size = 4`	grosor de la caja, NOTA: FUNCIONA CON "BW, CLASSIC, LIGHT,LINEDRAW Y TEST"

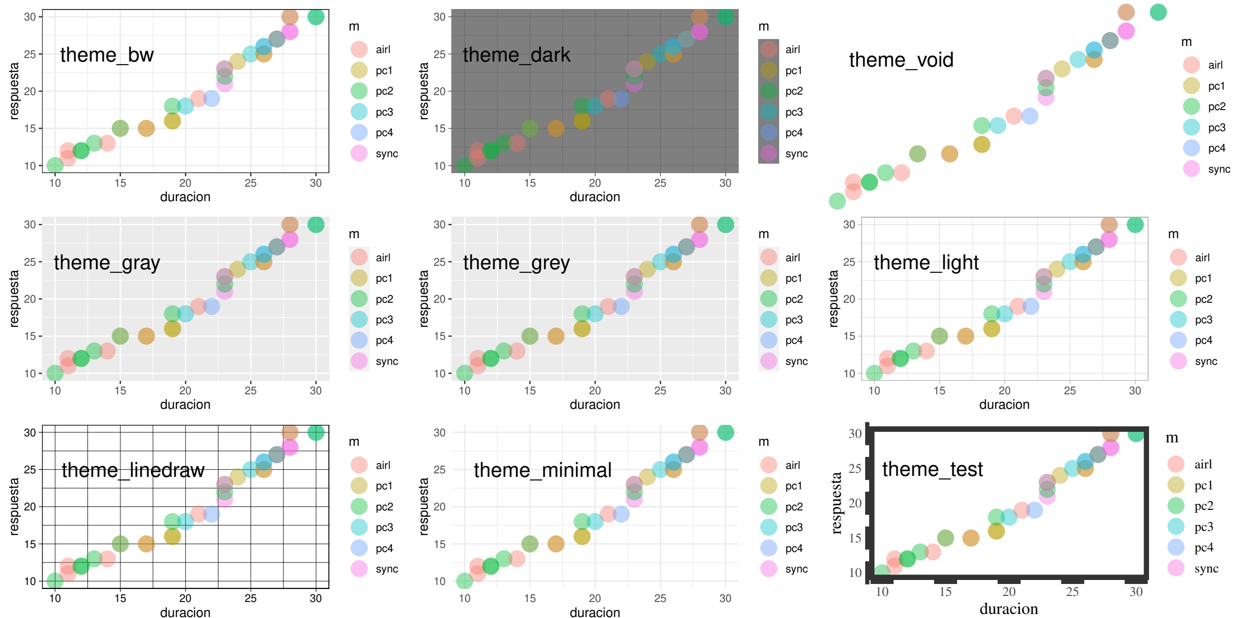
temas

```
#library(ggplot2)
#library(tibble)
# ver data tb
> q <- ggplot(data = tb,aes(x=duracion,y=respuesta,color=m))
> p <- q + geom_point(size=6,alpha=0.4)

> p0 <- p + theme_bw() +
    annotate("text",x = 14,y = 25,label="theme_bw", size= 6)
> p1 <- p + theme_classic() +
    annotate("text",x=14,y=25,label="theme_classic",size= 6)
> p2 <- p + theme_dark() +
    annotate("text",x = 14,y = 25,label="theme_dark",size=6)
> p3 <- p + theme_void() +
    annotate("text",x = 14,y=25,label="theme_void", size= 6)
> p4 <- p + theme_gray() +
    annotate("text",x = 14,y=25,label="theme_gray", size= 6)
> p5 <- p + theme_grey() +
    annotate("text",x = 14,y=25,label="theme_grey", size= 6)
> p6 <- p + theme_light() +
    annotate("text",x =14,y=25,label="theme_light", size= 6)
```

```
> p7 <- p + theme_linedraw() +
    annotate("text",x=16,y=25,label="theme_linedraw",size=6)
> p8 <- p + theme_minimal() +
    annotate("text",x=16,y=25,label="theme_minimal",size= 6)
> p9 <- p + theme_test(base_size = 14,
              base_family = "Times",
              base_line_size = 9,
              base_rect_size = 4) +
    annotate("text",x = 14,y=25,label="theme_test",size=6)

> gridExtra::grid.arrange(p0,p2,p3,p4,p5,p6,p7,p8,p9)
```



Otra función importante es la siguiente
`p + theme_bw() + facet_grid(rows=vars(m))`
las fuentes tipográficas que acepta ggplot son : “Short, Canonical, mono, Courier, sans, Helvetica, serif, Times, AvantGarde ,Bookman ,Helvetica-Narrow, NewCenturySchoolbook, Palatino, URWGothic, URWBookman, NimbusMon, URWHelvetica, NimbusSan, NimbusSanCond, CenturySch, URWPalladio, URWTimes, NimbusRom”

también la forma “face =” `"plain","bold","italic","bold.italic"`

Si en los ejes "x" e "y" muestran textos sobrepuestos, debe agregar esta función:

```
p + scale_x_discrete(guide = guide_axis(n.dodge = 2))
```

O esta otra función que elimina las etiquetas que unicamente están sobre montadas:

```
p + scale_x_discrete(guide = guide_axis(check.overlap = TRUE))
```

Si quieres centrar el titulo del gráfico "caption" agrega lo siguiente:

```
p + theme(plot.title = element_text(hjust = 0.5))
```

Mas información

En el libro **R Graphics Cookbook**[2], "capitulo 8" y sección "Fonts" usted puede encontrar más ejemplos sobre las fuentes tipográficas, temas y estilos.

CAPÍTULO 7

Importando y Exportando ficheros

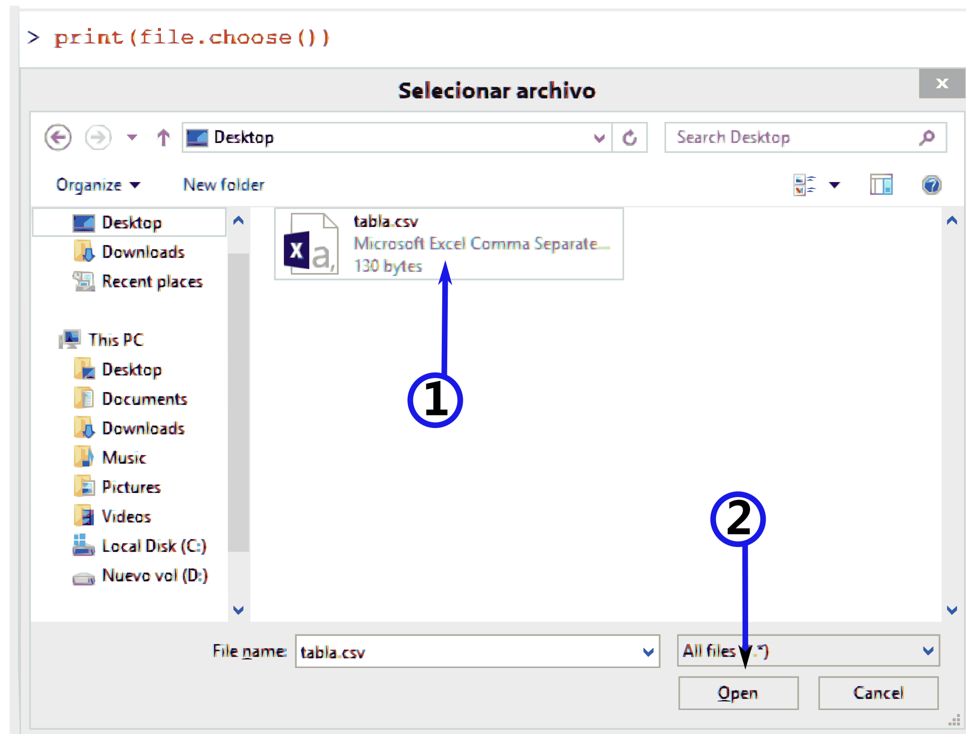
En toda esta sección se encontrará con la función **file.choose()** y bueno sirve para encontrar el fichero o la dirección relativa a ella por ejemplo:

🗁C:/Users/Den/Desktop/tabla.csv (Den: Usuario del equipo)

O sea que con esta función no se necesita escribir el directorio sino mas bien aparecerá una interfaz que permite recorrer de forma gráfica donde esta el fichero que se desea abrir y retornará una dirección, ¿muy interesante no te parece?. Ahora intenta esto:

```
> print(file.choose())
[1] "C:\\Users\\Den\\Desktop\\tabla.csv"
```

Figura 7.1: *Abrir fichero*



Si usamos nuevamente **file.choose()** como exportación de fichero entonces en **File name** se le da un nombre y el formato que se desea exportar por ejemplo "**exportar.csv**" y después presiona el botón **Open**, lo que generará otra ventana advirtiendo que dicho fichero no existe pero nos ofrece la opción de crear uno y es ahí donde presionas en el botón **Yes** para crearlo. Recuerda que esto no crea el fichero sino mas bien genera un enlace con el nombre del fichero a crear.

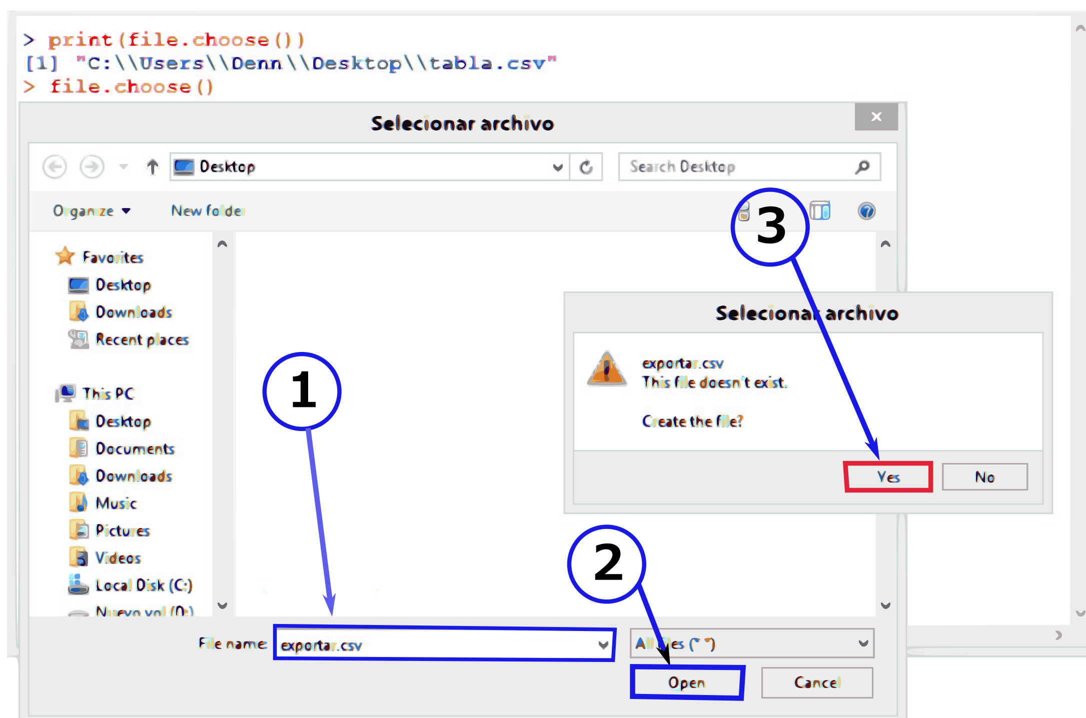
```
> file.choose()
[1] "C:\\Users\\Denn\\Desktop\\exporta.csv"
```

Revise los **ejemplos 2.1 y 2.2(pagina 62)** que es otra alternativa personalizable para abrir y guardar ficheros.

```
Abrir(nombre ="txt texto" ,filtro = ".txt")

Guardar(nombre ="txt texto" ,filtro = ".txt")
```

Figura 7.2: *Crear fichero*



1.1 CSV

El fichero sencillo de manejar en R y se presentará algunas funciones que se encargan de importar y exportar fichero csv. A continuación se mostrará una tabla que se tomará como ejemplo para procesar en R, al lado izquierdo se encuentra la vista de la tabla y al lado derecho esta los datos en un fichero llamado **tabla.csv**, intente crear un fichero desde el bloc de notas y guárdelo como tabla.csv o también puede hacerlo directamente de software MS Excel.

tabla

	title1	title2
f1	13	10
f2	11	8
f3	18	13
f4	21,4	19,4
f5	16	12
f6	"12"	
@		

tabla.csv

```
      ,title1   ,title2
f1    ,13       ,10
f2    ,11       ,8
f3    ,18       ,13
f4    ,"21,4"   ,"19,4"
f5    ,16       ,12
f6    ,"12"
@
```

1.1.1 Importar

Se presentarán algunas funciones que permiten la importación de este fichero.

interna

read.csv(file,header,sep,quote,dec,fill,comment.char,row.names)

En file también puede agregar url, acepta enlaces que inicien con `http://, https://, ftp://, o ftps://` y terminen en .gz, .bz2, .xz, o .zip (caso de ser comprimido) y .csv por ejemplo:
`file="http:// ..../tabla.csv"`

file	**(enlace)** Se coloca el directorio o el enlace donde esta el fichero.
header	**(booleano)** Indica si el fichero tiene o no en la primera fila los nombres de las columnas.
sep	**(char)** Que tipo de separador usa el fichero, dentro de los más conocidos "**,**" o "**;**" o "**t**" o "**|**" y otros.
quote	**(char)** Si hay comentarios que por lo general van entre comillas y para eliminarlo se usa `quote="\""` .
dec	**(char)** Dice que tipo de símbolo usa para decimales por ejemplo "**.**" o "**,**".**Solo funciona si toda la columna es numeric**
fill	**(booleano)** Si `fill=T` entonces las celdas vaciás se llenarán con valor **NA**, caso contrario (F) lo deja en blanco.
na.strings	**(char)** Si por casualidad no funciona fill, entonces intenta `fill=T,na.strings=""`.
comment.char	**(char)** Si `comment.char="L"`, buscará todos los elementos que contengan ese valor y lo eliminará. Solo aceptará un carácter.
row.names	**(numerico)** Si escribe `row.names=1`, estará diciendo que la primera columna contiene los nombres para todas las filas.

Ejemplo 1

```
> tabla <- read.csv(file=file.choose(),
                    header=TRUE,sep=",",
                    row.names=1)
> tabla
       title1 title2
f1     13          10
f2     11           8
f3     18          13
f4      21,4      19,4
f5     16          12
f6          12
@
> View(tabla) # vista del IDE
```

Se observa que en los **decimales ha usado la coma** y en la ultima fila todavía se encuentra el char(@).

Ejemplo 2

```
> tabla <- read.csv(file=file.choose(),
                    header=T, sep=",",
                    row.names=1, quote = "\"",
                    dec=",", fill = T,
                    na.strings="",
                    comment.char = "@")
> tabla
      title1 title2
f1      13.0   10.0
f2      11.0    8.0
f3      18.0   13.0
f4      21.4   19.4
f5      16.0   12.0
f6      12.0     NA
> View(tabla) # vista del IDE
```

Se ha logrado corregir los casos de la coma por punto gracias a `quote = "\"", dec=","`, eliminar la ultima fila con `comment.char = "@"` y añadir el relleno del <NA> con

```
fill = T, na.strings=""
```

readr

read_csv(file, col_names, comment, quote, n_max,skip, na)

En esta librería se tiene control sobre la cantidad de datos que usted desea extraer y también el control sobre que carácter sobreescribir cuando tiene una celda sin valores pero resulta dificultoso cuando se tiene decimales dentro de comillas.

file	**(enlace)** Se coloca el directorio o el enlace donde esta el fichero.
col_names	**(booleano)** Indica si el fichero tiene una columna de nombres.
comment	**(char)** Sirve para omitir elementos que contienen dicho valor escrito.
quote	**(char)** Si hay comentarios que por lo general van entre comillas y para eliminarlo se usa `quote="\""` .
n_max	**(numeric)** Limita la cantidad de valores. **No acepta cadena "c(1:3)"**
na	**(char)** Permite cambiar valores de elementos nulos o vacíos
skip	**(numeric)** Si coloca `skip=3` entonces eliminará filas desde el 1 hasta 3. **Acepta solo un valor, no cadena**

Librería importante para poder usar esta función:

```
> library(readr)
> # si hay error usar
> install.packages("readr")
> library(readr)
```

Recuerda conectarse a internet para descargar **install.packages("readr")** ya después usa solo **library(readr)**.

En el ejemplo siguiente se observa que tiene un error y éste es generado por los números que están en comillas, preste atención al delimitador(**,**) y al quote (`\"`)

Ejemplo

```
> tabla <- read_csv(file=file.choose(),
                    col_names = T,na = "NA",
                    quote = "\"",comment = "@") ❶
> tabla
# A tibble: 6 x 3
  X1    title1              title2
  <chr> <chr>                <dbl>
1 f1    13                      10
2 f2    11                       8
3 f3    18                      13
4 f4    "21,4\"  ,\"19,4"       NA  <−
5 f5    16                      12
6 f6    12                      NA
```

Es posible que genere errores como la columna X1 que no existe y los delimitadores que son debido a `f4 ,"21,4" ,"19,4"` al eliminar las comillas con `quote = "\""` solo nos quedaría 21,4 haciendo de ello una nueva columna para 4.

csvread

csvread(file, coltypes, header, nrows, verbose, delimiter, na.strings))

Y por ultimo hay una librería externa que ofrece leer de 10M a más líneas con una super velocidad usando poca memoria de R. Bueno esta librería se diferencia de las otras por su gran velocidad en procesar los datos pero eso no lo hace el único que usted pueda usarlo ya que por alguna razón no están incluidas estas funciones **quote** y **dec**.

También es importante agregar que cuenta con la función de **nrows**, permitiendo solo cargar una determinada cantidad de filas.

Si desea mas información, visite **https://cran.r-project.org/web/packages/csvread/csvread.pdf**

file	**(enlace)** Se coloca el directorio o el enlace donde esta el fichero.
coltypes	**(string)** Necesita saber que typo de datos hay en columnas, ejm: `"integer"`, `"double"`, `"string"`, `"long"`, `"longhex"`
header	**(boolean)** ¿La primera fila contiene los nombres para las columnas? (TRUE o FALSE)
nrows	**(numeric)** Indique la cantidad de lineas a leer (solo un valor), por ejm 30.
verbose	**(boolean)** Imprime la cantidad de lineas.
delimiter	**(char)** Indique el tipo de separador usado en el fichero.ejm: "**,**" o "**;**" o "**t**" o "**|**"
na.strings	**(char)** Permite cambiar valores de elementos nulos o vacíos, ejem :`"NA"`, `"na"`,`"NULL"`, `"null"`, `""`

Librería importante para poder usar esta función:

```
> install.packages("csvread")
> library(csvread)
```

Recuerda conectarse a internet para descargar **install.packages("csvread")** ya después usa solo **library(csvread)**.

Al igual que la librería anterior ésta también se tiene dificultades al leer el fichero pero se tratará de corregirlo con algunas funciones internas que posee el lenguaje R llamada `rownames,lapply`

Ejemplo

```
> # verificar tipos de datos para coltypes
> map.coltypes(file = file.choose(), header = T)
        X    title1    title2
"string" "string" "string"
> # cargar fichero
> tabla <- csvread(file = file.choose(),
                  header = T,delimiter = ",",
                  coltypes =
                  c("string","string","string") )
> tabla
         title1    title2
1 f1       13        10
2 f2       11         8
3 f3       18        13
4 f4     "21,4"   "19,4"
5 f5       16        12
6 f6         "12"   <NA>
7     @      <NA>   <NA>
> # nombre de filas
> rownames(tabla) <- tabla[,1]        1
> # eliminar 1era columna
> tabla[,1] = NULL                   2
> # eliminar ultima fila
> tabla = tabla[-7,]               3
> # eliminar comillas
> tabla[] <-lapply(tabla, gsub, pattern='"', replacement='')
 ↪  4
> # cambiar coma por punto
> tabla[] <-lapply(tabla, gsub, pattern=',', replacement='.')
 ↪  5
> tabla
      title1    title2
f1      13        10
f2      11         8
f3      18        13
```

```
f4        21.4     19.4
f5       16          12
f6            12   <NA>
> View(tabla) #--> ver por interfaz
```

1. como los nombres de las filas son "1,2,3,4,5,6,7", entonces **los cambio por la 1era columna**.
2. **elimino la 1era columna sobrante** que es "f1,f2,f3,f4,f5,f6,@"
3. **elimino la ultima fila** que solo contiene al @ como único elemento.
4. uso lapply para **detectar las comillas** y eliminarlos.
5. usando la misma función con el objetivo de **cambiar comas por puntos**.

2.1.2 Exportar

interna

Para la exportación de fichero csv en R es muy sencillo, solamente se requiere de algunos comandos y listo. Al igual que en el caso de importar fichero, se presentará por librerías y el motivo es nada mas que usted elija con cual se siente más cómodo ya sea por rendimiento u otras funciones configurables. A continuación se presentará algunas librerías populares.

write.csv(x, file, row.names, fileEncoding)

x	**(objeto)** Hace referencia al objeto creado y que contiene los datos.
file	**(string)** Se necesita el directorio donde se va a guardar el fichero y la extensión, ejemplo `file="D:/User/exportar.csv"`
row.names	**(boolean)** Pregunta si, ¿el objeto creado tiene nombre en las filas?
fileEncoding	**(string)** Debes indicar la codificación del fichero, los más usados `"latin1"`, `"UTF-8"`.

Ejemplo exportar 1 " write.csv()"

```
> dt = data.frame("x"=c(sample(1:15,size = 26,replace=T)),
                  "y"=c(runif(n = 26, min = 8, max = 26 )),
                  "z" = sample(30:65,size = 26,replace = T),
                  row.names =c( sample(LETTERS,size =
                   ↪  26,replace = F)) )

> write.csv(x=dt, file=file.choose(), row.names=T,
 ↪  fileEncoding="UTF-8")
```

Recuerda que **file.choose()** es una interfaz gráfica. Si el archivo existe este lo actualizará, por favor cuidado al usar.

readr

write_csv(**x**, **path**, **col_names**, **na**, quote_escape)

write_excel_csv(**x**, **path**, **append**, **col_names**, **na**, **delim**)

x	**(objeto)** Hace referencia al objeto creado y que contiene los datos.
path	**(string)** Se necesita el directorio donde se va a guardar el fichero y la extensión, ejemplo `path="D:/User/exportar.csv"`
col_names	**(boolean)** Pregunta si, ¿el objeto creado tiene nombre en las columnas?
na	**(string)** Si algún elemento esta vacío entonces esta función permitirá modificar ese valor.
quote_escape	**(string)** Esto funciona para los elementos que están entre comillas, permite modificar los valores.
append	**(boolean)** Si el fichero existe, entonces este al ser TRUE solamente actualizará los datos.
delim	**(string)** permite modificar el delimitador o separador de cada elemento. Por ejemplo `delim=","`

Lo primero es abrir la librería para poder usar esta función, así que ejecute lo siguiente:

```
> library(readr)
```

Si al ejecutar **library(readr)** genera un error, entonces significa que no has instalado, por favor usar `install.packages("readr")`

Ejemplo exportar 2 “ write_csv()”

```
> dt = data.frame("x"=c(sample(1:15,size = 26,replace=T)),
                  "y"=c(runif(n = 26, min = 8, max = 26 )),
                  "z" = sample(30:65,size = 26,replace = T),
                  row.names =c( sample(LETTERS,size =
                   ↪  26,replace = F)) )
```

```
> write_csv(x = dt,path = file.choose(),append = F,col_names
↪  = T,na = "",
            quote_escape = "double")

> write_excel_csv(x = dt,path = file.choose(),append =
↪  T,col_names = T,
                  na = "",delim = ",")
```

Recuerda que **file.choose()** es una interfaz gráfica. Si el archivo existe este lo va a actualizador de acuerdo a **append**, por favor cuidado al usar.

data.table

fwrite(x, file, sep, col.names, dec, na)

x	**(objeto)** Hace referencia al objeto creado y que contiene los datos.
file	**(string)** Se necesita el directorio donde se va a guardar el fichero y la extensión, ejemplo `file="D:/User/exportar.csv"`
col.names	**(boolean)** Pregunta si, ¿el objeto creado tiene nombre en las columnas?
na	**(string)** Si algún elemento esta vacío entonces esta función permitirá modificar ese valor.
sep	**(string)** Permite modificar el tipo de separador que se usará.
dec	**(string)** Si hay elementos tipo decimal, entonces **dec** permite cambiar el símbolo.

Al igual que la librería anterior, esta también es una librería externa que se necesita instalar primero para después incluir en el actual proyecto.

```
> install.packages("data.table")
> library(data.table)
```

Ejemplo exportar 3 " fwrite()"

```
#  crear tabla tipo data.frame
> dt  = data.frame("x"=c(sample(1:15,size = 26,replace=T)),
                   "y"=c(runif(n = 26, min = 8, max = 26 )),
                   "z" = sample(30:65,size = 26,replace = T),
                   row.names =c( sample(LETTERS,size =
                    ↪  26,replace = F)) )
# almacenar
> fwrite(x = dt,file = file.choose(),sep = ",",col.names = T,
 ↪  dec = ".")
```

La función **fwite** según la comunidad de R lo considera la más rápida al exportar ficheros.

2.2 SPSS

También conocido por su nombre completo “IBM SPSS Statistics”, software dedicado a estadísticos muy similar al lenguaje R pero con la diferencia de su costo. En R hay muchas librerías que cumplen con la función de importar y exportar, claro que cada una de ellas ofrece distintas opciones.

1.2.1 Importar

foreign

Se comenzará con una librería interna llamada “**foreign**”, esta librería ofrece no solamente abrir ficheros **.sav** sino diversas extensiones de otros softwares que en su gran mayoría no son libres , empezamos con:

read.spss(file, to.data.frame, use.value.labels, use.missings)

file	**(enlace)** Se coloca el directorio o el enlace donde esta el fichero.
to.data.frame	**(booleano)** Si es TRUE entonces convierte a data.frame, caso contrario será una lista.
use.value.labels	**(booleano)** Permite escoger si los elementos son tomados desde las etiquetas(valores) o se mantiene como numérico.
use.missings	**(booleano)** Si hay elementos vacíos entonces con **TRUE** lo rellena con **NA**

Esta librería esta incluida dentro del lenguaje R pero se necesita llamarlo o incluir en el trabajo actual para poder usarlo y para ello use lo siguiente:



```
> library(foreign)
# si deseas actualizar libreria
> install.packages("foreign")
```

Ejemplo importar librería 1 “ read.spss()”

```
> tabla = read.spss(file=file.choose(),to.data.frame=T,
                    use.value.labels=T, use.missings=T)
```

```
# al usar use.value.labels = T, estoy tomando los valores
# de las etiquetas.
```

haven

La siguiente librería llamada **haven** que esta dentro de **tidyverse**, ofrece no solamente abrir el fichero ".sav" sino mas bien su antecesor ".por". La librería tiene dos funciones "read_sav() y read_por()" pero se puede omitir estas dos funciones y usar de forma general "read_spss()" que se encargará de abrir por medio de la extensión que el fichero tenga.

Hablemos sobre que beneficios ofrece esta librería y considero que los importantes son:

- ✔ Permite seleccionar algunas columnas que usted desea.
- ✔ Permite seleccionar una cierta cantidad de datos.
- ✔ Al igual que la librería anterior, también permite el rellenado de elementos vacíos.

read_spss(file, user_na, col_select, n_max)

file	**(enlace)** Se coloca el directorio o el enlace donde esta el fichero (sav,por y zsav).
col_select	**(lista)** Permite seleccionar las columnas que se desea cargar. `col_select= c(1,3,5)`
n_max	**(numerico)** Colocar la cantidad de filas a procesar, acepta una longitud.
user_na	**(booleano)** Si hay elementos vacíos entonces con **TRUE** lo rellena con **NA**

DATO IMPORTANTE: LA LIBRERÍA ES EXTERNA Y SE TIENE QUE OBLIGATORIAMENTE INSTALARLO.

```
> install.packages("haven")
> library(haven)
```

Se necesita internet para instalar.

Ejemplo importar librería 2 " read_spss()"

```
> read_sav(file = file.choose(),
           user_na = T,        ❶
```

```
            col_select = c(1,3), (2)
            n_max = 4) (3)
```

En (1) se encargará de rellenar los elementos vacios con **NA**, en (2) quiero solamente incluir las columnas 1 y 3 en la importación, por ultimo en (3) solo necesito importar 4 filas.

Por ultimo una librería muy interesante llamada **readspss** por el nombre se deduce que solamente se encargar de abrir ficheros que solo pertenezcan al software IBM SPSS y claro dentro de las extensiones son ".sav, .zsav, .por y .sav encriptado (con contraseña)". Por el momento esta en desarrollo y se necesita de Rtools para su instalación por lo que hace un poco complejo y si usted quiere instalarlo entonces ingrese en esta url **https://github.com/JanMarvin/readspss**

2.2.2 Exportar

foreign

Se comienza con la librería **foreign** que ya ha sido conocido en importar ficheros, entonces ahora nos enfocaremos en como exportar ficheros para el software SPSS. La librería por defecto exporta dos ficheros y uno de ellos es un documento de texto en formato ".txt"(es posible que usted se halla topado con este tipo de fichero).

¿ Porque exporta un fichero ".txt"?
La razón es que ahí se almacena la base de datos o los elementos que son visibles en SPSS como Vista de Datos(Data View).

Lo que si debe tener cuidado es en el nombre ya que en si hay dos ficheros que el nombre debe ser el mismo pero la extensión no. Si por alguna razón los nombre de los ficheros son distintos entonces al abrir el archivo con la extensión ".sps" con el software SPSS te generará un error que tiene que ver con que no se ha encontrado el otro fichero. La manera correcta es por ejemplo:

- ✔ exportar.sps
- ✔ exportar.txt

write.foreign(df, datafile, codefile, package)

df	**(objeto)** Se coloca el objeto que se desea exportar, de preferencia data.frame
datafile	**(string)** Necesita la dirección donde se almacenará la base de datos, fichero con extensión (txt). `datafile="c:/datos.txt"`
codefile	**(string)** Al igual que datafile, éste debe tener el mismo nombre pero la extensión debe ser (sps). `codefile="c:/datos.sps"`
package	**(string)** Por tratarse unicamente del software SPSS usar `package="SPSS"`

NOTA IMPORTANTE: INCLUIR LIBRERÍA EN EL TRABAJO ACTUAL, SI YA EJECUTÓ ESTE CÓDIGO POR FAVOR OMITIR

```
> library(foreign)
# si deseas actualizar libreria
> install.packages("foreign")
```

Ejemplo exportar librería 1 " write.foreign()"

```
# la funcion es creada para generar 120 datos aleatorios
> al <- function(){
    return(sample(x = c(1:4),size = 120,replace = T) )
  }
# crear valores para los factores
> etiqueta = c("Nada Satisfecho","Poco Satisfecho",
               "Satisfecho","Muy Satisfecho")
# crear factor con etiquetas
> a <- factor(x = al(),levels = c(1:4),labels=etiqueta )
> b <- factor(x = al(),levels = c(1:4),labels=etiqueta )
> c <- factor(x = al(),levels = c(1:4),labels=etiqueta )
> d <- factor(x = al(),levels = c(1:4),labels=etiqueta )
#----------------------------
# crear dataframe con factores
#----------------------------
> tb <- data.frame("P1"=a,"P2"=b,"P3"=c,"P4"=d)
#---------------------
#       exportar SPS
#---------------------
> write.foreign(df = tb,datafile = file.choose(),
               codefile = file.choose(), package = "SPSS")
```

haven

Como se había hablado anteriormente de la librería externa llamada **haven**, también cuenta con una función que permite exportar a ".sav y .zsav". Para exportar cuenta con argumentos muy limitados como por ejemplo escoges si se exporta de forma comprimida (.zsav) o no, aquí toma encuenta la extensión que se esta nombrando. Por otro lado también acepta la fuente de los datos a exportar como "data.frame".

write_sav(data, path, compress)

data	**(objeto)** Se coloca el objeto que se desea exportar, de preferencia data.frame
compress	**(booleano)** Si `compress=FALSE` entonces usar la extension a ".sav", caso contrario se esta diciendo que use otra extensión ".zsav".
path	**(string)** Necesita la dirección donde se almacenará la base de datos, si `compress=TRUE` entonces cambiar la extensión a ".zsav"(referencia a comprimido), caso contrario solo usar ".sav"

NOTA IMPORTANTE: INCLUIR LIBRERÍA EN EL TRABAJO ACTUAL, SI YA EJECUTÓ ESTE CÓDIGO ANTERIORMENTE POR FAVOR OMITIR



```
> install.packages("haven")
> library(haven)
```

Ejemplo exportar librería 2 " write_sav()"

```
# la funcion es creada para generar 120 datos aleatorios
> al <- function(){
    return(sample(x = c(1:4),size = 120,replace = T) )
  }
# crear valores para los factores
> etiqueta = c("Nada Satisfecho","Poco Satisfecho",
               "Satisfecho","Muy Satisfecho")
# crear factor con etiquetas
> a <- factor(x = al(),levels = c(1:4),labels=etiqueta )
> b <- factor(x = al(),levels = c(1:4),labels=etiqueta )
> c <- factor(x = al(),levels = c(1:4),labels=etiqueta )
```

```
> d <- factor(x = al(),levels = c(1:4),labels=etiqueta )
#----------------------------
# crear dataframe con factores
#----------------------------
> tb <- data.frame("P1"=a,"P2"=b,"P3"=c,"P4"=d)
#----------------------------
> write_sav(data = tb,path = file.choose(),compress = F)
```

3.3 Excel

Por tratarse de un software muy popular sobre todo sencilla interfaz, sin embargo el único inconveniente seria su precio por usar o mejor dicho la licencia por equipo, estos precios varían de acuerdo a la versión del software.

Las versiones de Excel que almacenan la base de datos son dos ".xls" y ".xlsx", lo que seria de gran importancia conocer para poder usar de acuerdo a las librerías que ofrece la comunidad de R.

1.3.1 Importar

readxl

read_excel(path, sheet, range,col_names,na)

De la librería **tidyverse** quien tiene un montón de funciones que son muy útiles para la ciencia de datos, en esta ocasión nos enfocaremos en como importar ficheros de Excel. Esta librería ofrece también cambiar valores nulos o vacíos y la opción de la cantidad de elementos a cargar.

path	**(string)** Se coloca la dirección donde esta el archivo Excel.
sheet	**(string)** Hace referencia a la hoja donde esta los datos (se encuentra en la parte inferior), por ejemplo `"tabla","Hoja1"`.
range	**(string)** Se debe colocar los nombres de columna y numero que Excel usa para hacer referencia a los elementos, por ejemplo `range="B5:E5"`.
col_names	**(booleano)** Si es TRUE, estará diciendo que la primera fila contiene los nombres de las columnas.
na	**(string)** Sirve para rellenar elementos vacíos.

NOTA IMPORTANTE: INSTALAR PRIMERO LA LIBRERÍA

```
# libreria completa
> install.packages("tidyverse")
# instalar solo para excel
> install.packages("readxl")
#
```

Ejemplo importar usando " read_excel()"

```
#  abrir libreria
> library(readxl)
#
#
> read_excel(path = file.choose(),
             sheet = "tabla",
             range = "B4:F10"
             col_names = T)
```

También se puede reducir el código si solo usamos range en vez de sheet. Solamente se necesita el nombre de la hoja de datos, seguido un signo de admiración y luego escribimos el rango.

```
> read_excel(file.choose() ,
             range = "tabla!B4:F10"
             col_names = T)
```

2.3.2 Exportar

writexl

write_xlsx(x, path, col_names,format_headers)

Se presenta una nueva librería llamada **writexl**, su única función es escribir un archivo xlsx para versiones superiores de Excel 2010, así que será de mucha utilidad si solo quieres exportar tu base de datos de R a Excel.

x	**(objeto)** Se coloca el objeto que tiene la base de datos.
path	**(string)** Colocar la dirección donde deseas guardar el archivo, ejemplo `path="c:/carpeta/exp.xlsx"`.
col_names	**(booleano)** Si es TRUE, estará diciendo que la primera fila contiene los nombres de las columnas.
format_headers	**(booleano)** Si en `col_names=T` y `format_headers=T`, estarás permitiendo que se escriba los nombres **en negrita y centrado**.

Nota importante: Instalar primero la librería

```
#  Instalar libreria
> install.packages("writexl")
#
#    abrir libreria
> library(writexl)
```

No se exporta los nombres de las filas "**row.names**" a Excel.

Ejemplo exportar usando " write_xlsx()"

```
# crear nueva base de datos
#
> dt = data.frame("x"=c(sample(1:15,size = 26,replace=T)),
                  "y"=c(runif(n = 26, min = 8, max = 26 )),
                  "z" = sample(30:65,size = 26,replace = T),
```

```
                  row.names =c( sample(LETTERS,size =
                   ↪  26,replace = F) ) )
#
#  Exportar
> write_xlsx(x = dt, path = file.choose(),
              col_names = T,
              format_headers = T)
```

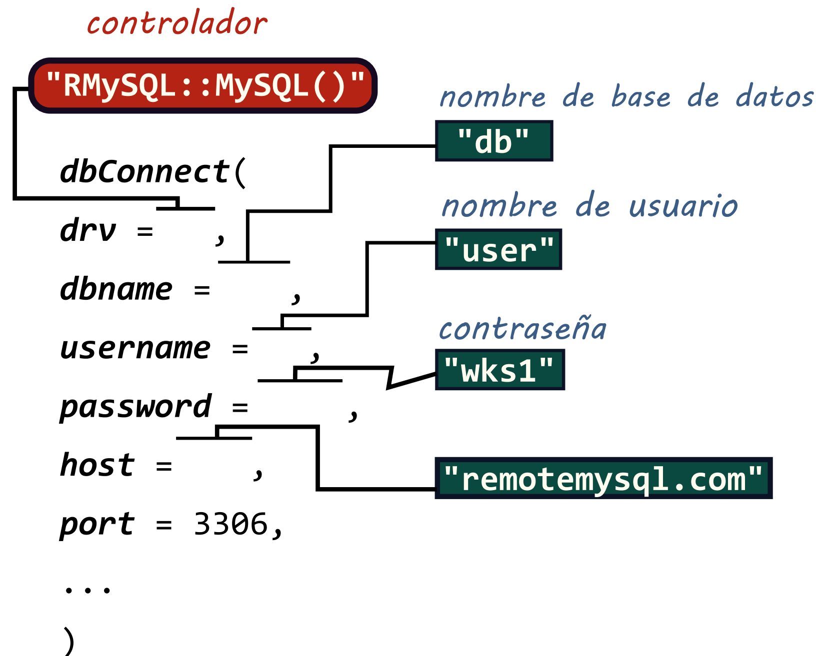
CAPÍTULO 8

Base de Datos

1.1 MYSQL

Como dice wikipedia. "MySQL es un sistema de gestión de base de datos relacional (RDBMS) de código abierto, basado en lenguaje de consulta estructurado (SQL)". Debido a su gran utilidad en trabajos, **R** ha decidido incluir una librería llamada **RMYSQL**.

Figura 8.1: *COMO CONECTAR EN RMYSQL*



Nota: los valores dados son un ejemplo, usted trate de cambiarlos

En la página **https://remotemysql.com/** ofrece un servidor remoto con 100MB de espacio libre y además se puede crear tantas tablas como se desea. En este libro se tomará como referencia esta web para simular una conexión remota no local.

Figura 8.2: *COMO REGISTRARSE EN remotemysql*

Create Account

Home Tutorial About

FREE MYSQL

Get a free remote MySQL database

Learn more

Login

Login | Create Account | Forgot Password?

Email *

Password *

Repeat Password *

3 + 8 *

By signing up to use this service you agree to our Terms and Conditions

Create Account

Una vez creada la cuenta, use login e inicie sesion

[redacted]@hotmail.com

DASHBOARD

DATABASES

STATISTICS

PRIVILEGES

LEARN MYSQL

Databases

All your databases are listed below. Create news ones, upto a maximum of 3, with the button at the bottom.

USERNAME | SIZE | CREATED DATE | ACTION

CREATE NEW DATABASE

USERNAME	SIZE	CREATED DATE	ACTION
LS6lpCIKMv	0.02 MB	2020-03-13 00:44:53	≡ ACTION

Statistics

Permissions

Reset Password

Delete

phpMyAdmin

CREATE NEW DATABASE

por favor guarde los datos:
Username:
Database:
Password:

Nombre de la tabla: TABLA Agregar 4 columna(s) Continuar

Estructura

Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento	Atributos	Nulo	A_I
ID	INT	11	Ninguno			☐	☑
Edad	INT	11	Ninguno			☐	☐
Genero	VARCHAR	10	Ninguno	latin1_spanish_ci		☐	☐
Vota	VARCHAR	10	Ninguno	latin1_spanish_ci		☐	☐

Previsualizar SQL Guardar

Bien, para comenzar se necesita instalar lo siguiente:

```
> install.packages("RMySQL")
        ⋮
# library(DBI)
> library(RMySQL)
```

1.1.1 Conectar

Una vez registrado en la página web y obtenido los datos como **Username**, **Database** y **Passsword**. Entonces nos conectamos de la siguiente manera:

```
> con = dbConnect(RMySQL::MySQL(), dbname="LVlpCJKMv",
↪  username="LVlpCJKMv",
          password="I6xLtEGSQt", host="remotemysql.com")
```

Importante: La conexión dura muy poco así que por cada 5 minutos intentar nuevamente ejecutar este objeto "**con**", caso contrario no se podrá hacer consultas.

Como nuestra tabla se llama "**TB**" y queremos saber los elementos que contiene, bueno en **R** hay dos maneras de hacer este proceso: mediante consulta (**dbGetQuery**) o por una función sencilla(**dbReadTable**). Ambos retornan una clase **tipo data.frame**.

✓dbReadTable

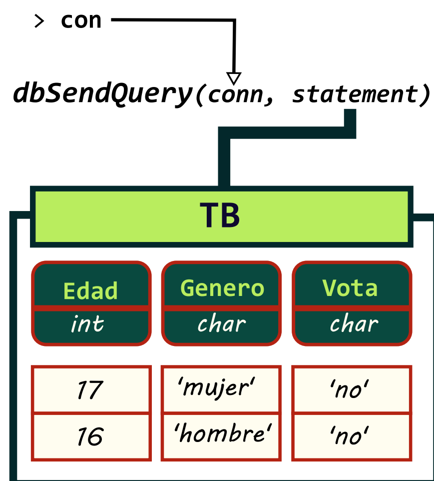
```
> tb=dbReadTable(con,"TB")
# -----> verificar <----
> tb
[1] ID  Edad Genero Vota
<0 rows> (or 0-length
 ↪  row.names)
> class(tb)
[1] "data.frame"
```

✓dbGetQuery

```
> sql <- "SELECT * FROM TB"
> tb = dbGetQuery(con,sql)
# ----> verificar <-----
> tb
[1] ID Edad Genero Vota
<0 rows> (or 0-length
 ↪  row.names)
> class(tb)
[1] "data.frame"
```

La tabla "**TB**" no tiene ningún elemento, únicamente genera un reporte indicando que se tiene 0 filas. Si deseas agregar elementos a la base de datos remota, en RMySQL hay una función que funciona contrario al **dbGetQuery** y que es llamado **dbSendQuery(con,consulta)**.

2.1.2 Insertar



Intentaremos crear un objeto "**sql**"con el objetivo de escribir la sintaxis SQL y que más después nos servirá para hacer consultas. La función más efectiva en **R** es la llamada **paste()**.

> **Nota:** Primero debes saber que tipo de valores recibe cada columna: **Edad**(numerico),**Genero**(char),**Vota**(char)

```
"INSERT INTO TB() VALUES()"
```

Edad(int)

```
> tmp <- "INSERT INTO TB(Edad)"
> lb <- paste(tmp,"VALUES(",tab$x[1],")",sep="")
> lb
[1] "INSERT INTO TB(Edad)VALUES(34)"
```

Genero(char)

```
> tmp <- "INSERT INTO TB(Genero)"
> lb <- paste(tmp,"VALUES('",tab$y[1],"')",sep="")
> lb
[1] "INSERT INTO TB(Genero)VALUES('hombre')"
```

Vota(char)

```
> tmp <- "INSERT INTO TB(Vota)"
> lb <- paste(tmp,"VALUES('",tab$z[1],"')",sep="")
> lb
[1] "INSERT INTO TB(Vota)VALUES('no')"
```

Has notado que tanto los valores de **Vota** como **Genero** van entre comillas simples y esto es simplemente porque ambos reciben solo caracteres (char), esto es importante ya que si no lo usas y ejecutas el comando en SQL es muy probable que te genere un error.

Se intentará generar valores aleatorios en **R** usando la super función **sample()** que además funciona con caracteres.

```
> al <- function(i = 1){
       edad <- sample(x = c(20:54),size = i, replace = T)
       genero <- sample(x = c("hombre","mujer"),size =
        ↪  i,replace = T)
       vota <- sample(x = c("si","no"),size = i,replace = T)
       return(data.frame("x"= edad,"y"=genero,"z"= vota))
    }

> tab <- al(i = 30)
```

Usaremos la siguiente función siguiente para enviar los 30 datos:

✓dbSendQuery(conn, statement)

```
for (i in seq_len(nrow(tab))){
    sql = "INSERT INTO TB(Edad,Genero,Vota) VALUES("
    lb = paste(sql,tab$x[i],"," ,"'",tab$y[i],"'" ,","
     ↪  ,"'",tab$z[i],"'",")" )
    dbSendQuery(con,statement = lb)
}
```

El primer valor que contiene **lb** es:

```
[1]
 ↪  "INSERT INTO TB(Edad,Genero,Vota) VALUES(34,'hombre','no')"
```

Para ver si ha procedido con la subida de los datos

Nota: recuerda ejecutar el objeto **con** por si genera un error similar a `object 'con' not found`

```
> tabla <- dbReadTable(conn = con,name = "TB")
> tabla
   ID Edad  Genero Vota
1   1   51   mujer   no
2   2   51  hombre   no
⋮   ⋮   ⋮      ⋮     ⋮
20 20   51  hombre   si
⋮   ⋮   ⋮      ⋮     ⋮
29 29   23  hombre   si
30 30   34   mujer   si
```

3.1.3 Editar

Como en el caso anterior que usamos **dbSendQuery**, también nos sirve para editar un elemento de una determinada tabla ya que se esta hablando de hacer una consulta y solamente se cambiará “INSERT INTO” por “UPDATE .. SET”.

```
"UPDATE TB SET col1=?, col2=?, WHERE ID=?;"
```

Por ejemplo quiero cambiar la fila 24 que contiene

> **Nota:** recuerda ejecutar el objeto **con** por si genera un error similar a `object 'con' not found`

```
> dbGetQuery(conn = con,"SELECT * FROM TB WHERE ID = 24 ")
  ID Edad  Genero Vota
1 24   49  hombre   si
```

Intentaremos hacerlo con otra función que simplifica lo de **paste()** en lo que respecta al gran espacio que se ocupa para definir la mezcla entre objetos y caracteres.

sprintf("%s %s %s ", uno,dos,tres)

```
> ed = 54
> gen = "mujer"
> vota = "si"
> sq = "UPDATE TB SET "
> sql = sprintf("%s Edad=%s,Genero='%s',
                    Vota='%s' WHERE ID=24",
                    sq,ed,gen,vota)
> dbSendQuery(conn = con,statement = sql)
<MySQLResult:569974448,3,0>
```

Verificamos

```
> dbGetQuery(con,"SELECT * FROM TB WHERE ID = 24 ")
  ID Edad Genero Vota
1 24   54  mujer   si
```

4.1.4 Eliminar

La sintaxis para eliminar un determinado elemento en sql es el siguiente

```
"DELETE FROM TB WHERE condición;"
```

Por ejemplo quiero eliminar la fila 24 y eso seria así

```
> dbSendQuery(con,"DELETE FROM TB WHERE ID = 24")
<MySQLResult:569974448,4,0>
```

Verificamos

```
> dbGetQuery(con,"SELECT * FROM TB WHERE ID>21 AND ID<27")
  ID Edad  Genero Vota
1 22   52   mujer   si
2 23   39  hombre   si
3 25   53  hombre   si
4 26   52   mujer   no
```

Si intentas eliminar una columna, usa lo siguiente:

```
"ALTER TABLE TB DROP COLUMN col;"
```

TB Nombre de la tabla

col Nombre de la columna

> **Importante:** La columna que intentas eliminar no debe tener relacion con otra tabla o sino te generaría un error

Pero si su caso es que quiere eliminar la tabla por completo, use

```
"DROP TABLE TB;"
```

O solamente elimine el contenido de la tabla pero no la tabla en si

```
"TRUNCATE TABLE TB; "
```

CAPÍTULO 9

Creando Reportes

1.1 MS Word

Existe una librería llamada **officer** que permite generar un archivo **.docx** compatible con las versiones actuales de Word y lo mejor es que no es necesario instalar el software Microsoft office. Es una librería externa y así que se necesita instalar lo siguiente.

```
> install.packages("officer")
```

Es recomendable tener la versión de **R** actualizado

Algunas funciones muy importantes en officer:

Cuadro 9.1: *Officer funciones*

función	descripción
read_docx()	iniciar documento
body_add_par(x,“ ”,style,pos)	agregar párrafos
body_add_table(x,value,style)	agregar tabla
body_add_gg(x,value,width,height,style)	agregar gráficos (solo ggplot)
body_add_img(x,src,width,height,style)	agregar gráficos externos
⋮	⋮

Algunos estilos comúnmente usados

Cuadro 9.2: *Officer style*

style	descripción
"Normal"	texto normal
"heading 1"	primer subtitulo
"heading 2"	segundo subtitulo
"heading 3"	tercer subtitulo
"table_template"	estilo 1 en tablas
"Table Professional"	estilo 2 en tablas
"Light List Accent 2"	estilo 3 en tablas
"table title"	titulo para tablas
"graphic title"	titulo para gráficos
⋮	⋮

Si usted tiene instalado Rstudio intente crear un nuevo fichero R script tecleando (ctrl + ⇧ + N) o desde su IDE preferido del lenguaje R trate de crear un nuevo fichero R script, guárdelo. Una vez que tiene un fichero "documento.R" dentro de él escriba lo siguiente:

```
#   librerias
#-------------------------------
> library(officer)
> library(magrittr)
> library(ggplot2)
> library(plotrix)
#-----------------
> directorio <- "D:/reporte/word/"
#------------------------------- ❶
> sp <- c(1,1,1,4,4,4,4,4,4,4,3,3,3,3,3,2,
        2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2)
> indicador <- factor(x = sp ,levels = c(1:4),
             labels = c("Nada Satisfecho",
                        "Poco Satisfecho",
                        "Satisfecho",
                        "Muy Satisfecho"))
> tab <- as.data.frame(table(indicador))
#------------------------------- ❷
> grafico3 <- ggplot(tab,aes(x="",y=tab$Freq,
                   fill=factor(tab$indicador))) +
```

```
  geom_bar(width = 1,stat = "identity") +
  coord_polar("y") +
  geom_text(aes(label =
                  paste(round(Freq/sum(Freq)*100,1),"%")),
                       position=position_stack(vjust = 0.5))+
  theme_void(base_size = 18)+labs(fill="Indicador")
#------------------------------- 3
> fig <- paste(directorio,"figura.wmf",sep = "")
> ggsave(plot = grafico3,filename = fig,device = "wmf",
                                   dpi = 72,limitsize = T)
#------------------------------- 4
> color <- c("#526189","#f6f6f6","#d3d9e7","#a3acc5")
#-----------
> fan <- paste(directorio,"figura2.wmf",sep = "")
> win.metafile(file = fan,height = 5,width = 10,
                                        pointsize = 14)
> fan.plot(x = tab$Freq,labels=as.character(tab$indicador),
           col = color,ticks = 30)
> dev.off()
#-------------------------------
> parrafo = " Se ha hecho una investigación en una empresa
          que cuentra con 32 empleados, cuyo objetivo es
          saber si estan satisfechos en su desempeño laboral
          dentro de la empresa:"
#-------------------------------
> guardar <- paste(directorio,"archivo.docx",sep="")
> docu <- read_docx()
> docu <- docu %>%
    body_add_par("Reporte ...",style = "heading 1") %>%
    body_add_par("Satisfacción Laboral",
                             style = "heading 2") %>%
    body_add_par(parrafo,style = "Normal") %>%
    body_add_table(tab,style = "Light List Accent 2") %>%
    body_add_gg(docu, value = grafico3,
                style = "centered",width=5,height=5) %>%
    body_add_par(value = "Grafico Pie",
                            style = "graphic title") %>%
    body_add_par("",style = "Normal") %>% #-- nueva linea --
    body_add_par(parrafo,style = "Normal") %>%
    body_add_par("Amabilidad en su atención",
                               style = "heading 2") %>%
```

```
        body_add_img(src = fan,height = 3,
                                  width = 6,pos = "after"  )%>%
        body_add_par(value = "Grafico abanico",
                                   style = "graphic title") %>%
        body_add_par("",style = "Normal") %>% #-- nueva linea --
        body_add_par(parrafo,style = "Normal") %>%
        body_add_par("Representación personal",
                                         style = "heading 2")
  > print(docu,target = guardar )
```

En 1 solamente se crea la data, con la ayuda de **table()** se ha podido sumar de acuerdo a la escala dada(1,2,3,4).

En 2 esa misma data "**tab**" se ha hecho un gráfico de torta usando ggplot. Presta atención en **body_add_gg()**

En 3 se usa **ggsave** para exportar gráfico y después se obtiene la referencia (**fig**)
body_add_img(src=fig) para mayor calidad al exportar.

En 4 es igual al 3 por el motivo que se tendrá que exportar la gráfica y dicha dirección se almacena en el objeto **fan** para después usarlo en **body_add_img(src= fan)**

En el último comando `print(docu, target)` en "target" se coloca el directorio donde se guardará el fichero ".docx", por ejemplo
`target="D:/reporte/word/archivo.docx"`

Figura 9.1: *Diseño al compilar A*

1. Reporte ...

1.1. Satisfacción Laboral

Se ha hecho una investigación en una empresa que cuentra con 32 empleados, cuyo objetivo es saber si estan satisfechos en su desempeño laboral dentro de la empresa:

indicador	Freq
Nada Satisfecho	3
Poco Satisfecho	17
Satisfecho	5
Muy Satisfecho	7

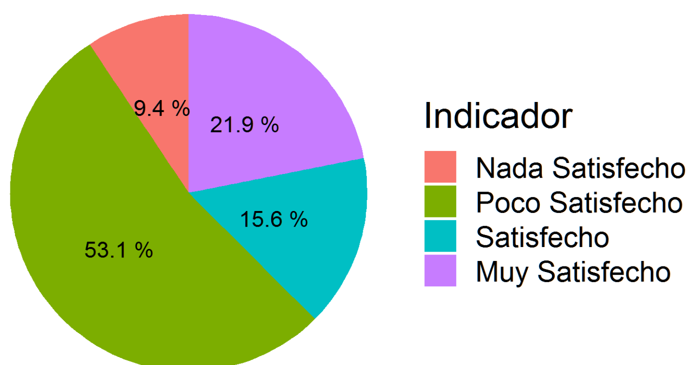


Grafico Pie

Se ha hecho una investigación en una empresa que cuentra con 32 empleados, cuyo objetivo es saber si estan satisfechos en su desempeño laboral dentro de la empresa:

Figura 9.2: *Diseño al compilar B*

1.2. Amabilidad en su atención

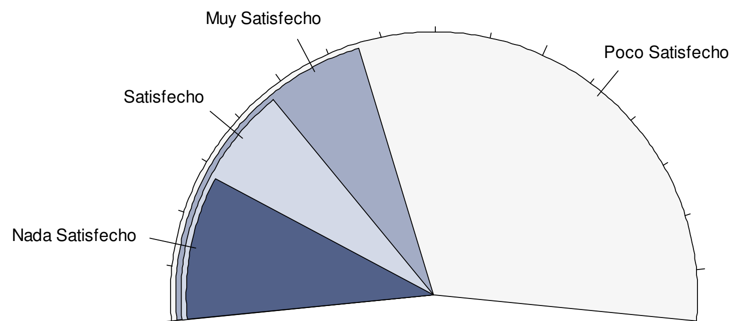


Grafico abanico

Se ha hecho una investigación en una empresa que cuentra con 32 empleados, cuyo objetivo es saber si estan satisfechos en su desempeño laboral dentro de la empresa:

1.3. Representación personal

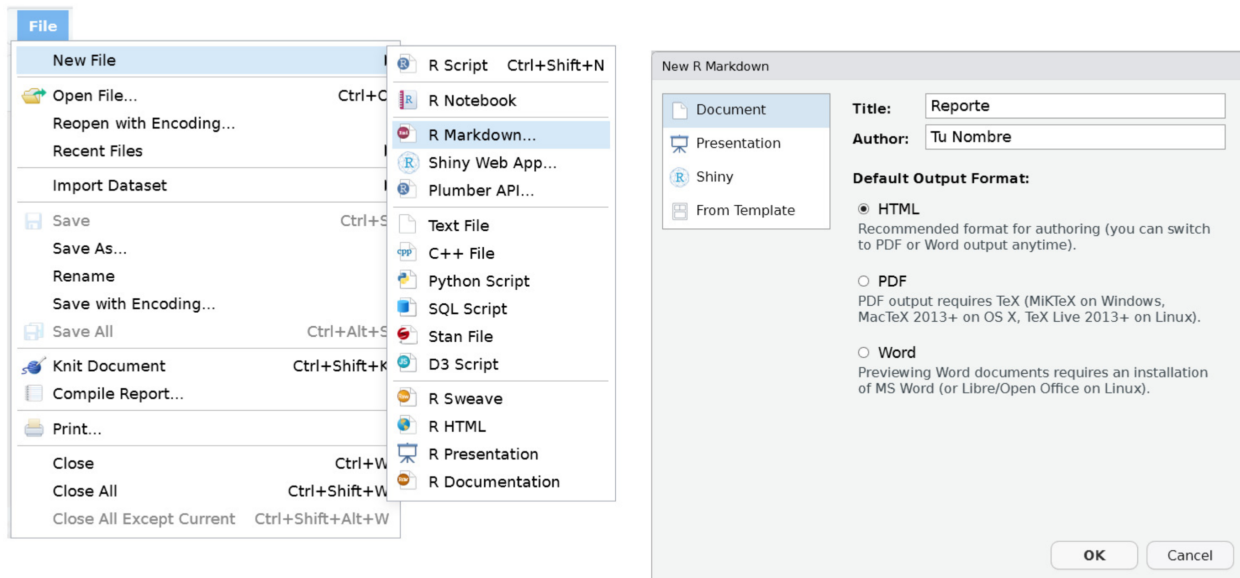
2.2 Markdown

Para crear reportes de tipo Markdown con R, es 100 % recomendable usar el software Rstudio por el único motivo que tiene compatibilidad con pandoc y este a su vez convierte a formatos como "html, pdf y word". ¿Genial no te parece?

Debo hacerte recordar que la exportación a word no es tan buena, en especial las imágenes que no son en alta calidad.

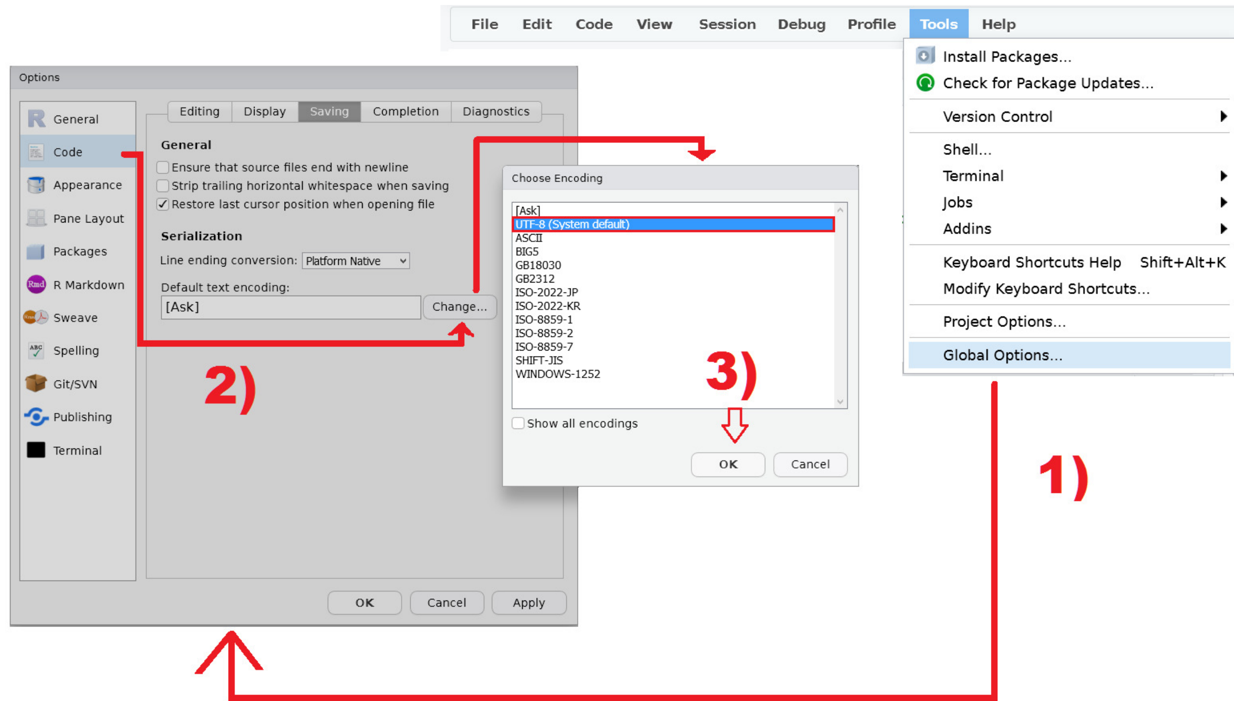
Si estas en esta unidad y **aún no has instalado los paquetes en MikTex**, entonces regresa al capitulo 1 "Instalación" y la subsección "**Softwares Externos**", es importante que hagas este paso para no tener errores de compilación. Si ya lo tienes entonces inicie Rstudio y crea un documento Rmd(ver figura 9.3)

Figura 9.3: *Crear fichero Rmarkdown desde Rstudio*



Otro dato importante es que después de guardar el fichero con (ctrl + S), intente abrirlo en codificación utf-8 para que así permita reconocer texto español como tilde y otros casos como la eñe.

Figura 9.4: *Codificación del fichero global*



Rstudio convierte el archivo .Rmd a pdf mediante el uso de Pandoc y LaTeX.

Como todo documento, siempre tiene su cabecera o la configuración que permite definir detalles como:

Cabecera

```
---
title: "Reporte"
author: "Tu nombre"
date: '2020-03-12'
header-includes:
  - \usepackage[spanish]{babel}
output:
  pdf_document:
    df_print: kable
    fig_caption: yes
    toc: yes
documentclass: article
classoption: 12pt,letter,twocolumn
geometry: margin=1in
lang: es-ES
---
```

documentclass Clase de documento (article,book,report,memoir,beamer)

classoption hay tres:

tamaño de letra 10pt,11pt y 12pt.

tamaño de papel a4paper,a5paper,legalpaper ...

tipo onecolumn y twocolumn.

geometry la geometría del papel y margenes admitidos(top,left,right), recomiendo dar una revisada a su documentación **http://ctan.dcc.uchile.cl/macros/latex/contrib/geometry/geometry.pdf** y también en el capitulo 9 sección 3.3 de este libro.

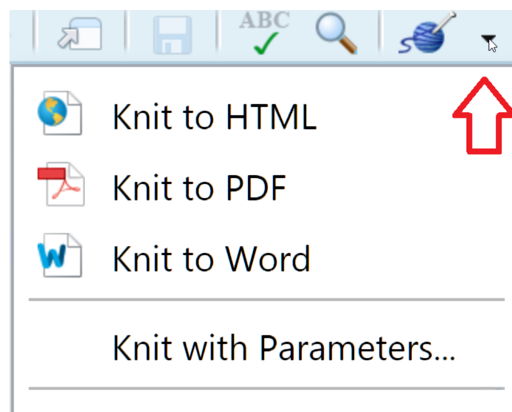
header_includes Incluir paquetes, si desea que los títulos de las tablas y de figuras estén en la parte superior o top en ingles, agrega a continuación: `\usepackage{floatrow}` , después `\floatsetup[figure]{capposition=top}` y por último `\floatsetup[table]{capposition=top}`

Todo lo que esta dentro de **pdf_document** permite tanto como incluir toc(tabla de contenidos) como títulos en gráficos y tipo de tabla.

Cuando agregas "showframe" en `geometry: margin=2cm,showframe`, te permite ver los márgenes del documento.

RMarkdown ofrece compilar varios formatos como son: HTML, PDF Y Word

Figura 9.5: *Export*





Generar PDF

La exportación a pdf no funcionará si no tienes instalado los softwares **MiKTeX** y **pandoc** (descargar ultima versión). Si estas en Windows, ambos deben estar en las variables de entorno (path and environment).Revise el tema instalación cap 3.3

1.2.1 Texto Normal

Los saltos de líneas o líneas vacías juegan un papel muy importante en Markdown, tales permiten separar y crear párrafos. Markdown resulta ser mas sencillo para crear artículos rápidos pero sino respetas ciertas reglas pueda que te compliques.

secciones o encabezados

Los encabezados se dividen en seis niveles:

Cuadro 9.3: *Lista de Encabezados*

Escribir	**Produce**
`# Encabezado 1`	Encabezado 1
`## Encabezado 2`	Encabezado 2
`### Encabezado 3`	Encabezado 3
`#### Encabezado 4`	Encabezado 4
`##### Encabezado 5`	Encabezado 5
`###### Encabezado 6`	Encabezado 6

estilos en caracteres

Para usar itálica simplemente se usa un asterisco "**`*texto en itálica*`**" o un guión bajo "**`_texto en itálica_`**" pero si al asterisco se le agrega otro asterisco, esto pasa a ser negrita "**`**texto en negrita**`**".

También esta el subrayado y LaTeXofrece su comando "**`$\underline{texto subrayado}$`**".

Cuadro 9.4: *Lista de estilos*

Escribir	**Produce**
`*itálica*`	*itálica*
`**negrita**`	**negrita**
`**_itanegrita_**`	***itanegrita***
`$\underline{subrayado}$`	subrayado
`~~tachado~~`	~~tachado~~

```
**Modo Html**

Este es un  texto en *_itálica_* pero a la vez es <br/>
**negrita** si se agrega otro `*`. Además puedo usar <br/>
ambos a la vez **_itálica en negrita_** <br/>

el `<br>` es para saltos de linea <br/>
en Html <br/>

**Modo Latex pdf**

también puedes usar \newline
otra linea
```

Modo Html

Este es un texto en *itálica* pero a la vez es **negrita** si se agrega otro `*`. Además puedo usar ambos a la vez ***itálica en negrita***

el $< br >$ es para saltos de linea en Html

Modo Latex pdf

también puedes usar
otra linea

listas

Si estas en un párrafo o cualquier otro caso, da un salto de línea para crear la lista. Las listas se crean mediante lo siguiente: un espacio seguido de un guión(número o letra) seguido de un espacio y después el texto que deseas en esa lista, por ejemplo:

```
párrafo cualquiera
 - lista a
 - lista b
   - lista a
     - lista b
```

Los cuadrados representa la separación de espacios creados con la barra espaciadora, es muy importante que hagas esto

Listas
```
Digamos un párrafo cualquiera

 - lista a
 - lista b
   - lista b1
     - lista b1.1

Otra lista

 a. lista 1
 b. lista 2
   - lista b1
     - lista b1.1
```

Digamos un párrafo cualquiera

- lista a
- lista b
  - lista b1
    - lista b1.1

Otra lista

a. lista a

b. lista b
  - lista b1
    - lista b1.1

```
¿Le agrada R Markdown?

 - [x] Si
 - [ ] No
```

¿Le agrada R Markdown?

☒ Si

☐ No

También esta la citas en bloques, presta atención al comando `<br>` que en html es un salto de linea y LaTeX es `\par`. Si exportas a documento pdf, para saltos de linea reemplaza `<br>` por `\par`:

```
> Cita en bloque <br>
continua en <br>
otra linea...

> Cita
> en bloque
```

> Cita en bloque
> continua en
> otra linea ...

> Cita en bloque

notas al pie de página

Esta función también muy importante al crear un documento y consiste en usar `[^1]` para hacer referencia:

```
los otros [^1] también se impri...

[^1]:son java,php,Ruby,css
```

los otros [1] también se impri...

[1] son java,php,Ruby,css

2.2.2 texto matemático

Hay varios símbolos que permiten imprimir textos matemáticos, por ejemplo si deseo incluir un símbolo matemático en la misma linea: **α**, esto generará el símbolo α en la misma linea.

```
Con un $\pmb{\alpha}= 0.05$ se ...

\[
 a^{2} = 2x+y
\]

Aquí otra forma:

$$
 n = \frac{2}{3}
$$
```

Con un $\boldsymbol{\alpha} = 0{,}05$ se ...

$$a^2 = 2x + y$$

Aquí otra forma:

$$n = \frac{2}{3}$$

El símbolo **\pmb{}** permite convertir unicamente el texto matemático en negrita.

3.2.3 código

Existe una lista de infinidades lenguajes de programación y Rmarkdown también cuenta con la función de incluir la sintaxis de dicho lenguaje para ser mostrado en el documento.

````
En una linea va un  `codigo` simple

```python
import statistics as stad

tr = [32, 13, 25, 36, 56, 24, 11, 28, 45, 31,23,38]
````
````

````
print(stad.quantiles(tr))
#[23.25, 29.5, 37.5]
```

```html
 <button type="button">Boton</button>
```
````

En una linea va un `codigo` simple

```python
import statistics as stad

tr = [32, 13, 25, 36, 56, 24, 11, 28, 45, 31,23,38]
print(stad.quantiles(tr))
#[23.25, 29.5, 37.5]
```

```html
<button type="button">Boton</button>
```
```

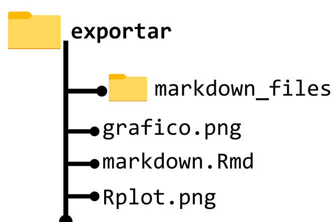
4.2.4 figuras

Librerías R importantes que se deben de incluir dentro del documento:

Figura 9.6: *Librerias importantes RMarkdown*

````
```{r ,warning=FALSE}
library(knitr)
library(ggplot2)
library(plotrix)
```
````

y en el directorio se encuentra así



locales

En el directorio tengo dos imágenes: grafico.png y Rplot.png
Intentaré usar esas imágenes para incluir en el documento actual:

````
```{r img, echo=TRUE,out.width="45%",fig.cap="grafico",
          fig.pos="!htbp",fig.align='center'}

include_graphics("grafico.png")

```

hago referencia a la figura \ref{fig:img}
````

lo que se ha hecho es aplicar reducción del tamaño de la imagen de manera proporcional (**out.width**) y después la posición precisa donde quiero que se encuentre (**fig.pos**).



Para mas información por favor visite la **tabla 9.21**

creadas

Aquí se necesitarán las librerías pero primero intentemos crear nuestro marco de datos y se muestra a continuación:

```
```{r, echo=FALSE}
sp <- c(1,1,1,4,4,4,4,4,4,4,3,3,3,3,3,2,
        2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2)
indicador <- factor(x = sp ,levels = c(1:4),
             labels = c("Nada Satisfecho",
                        "Poco Satisfecho",
                        "Satisfecho",
                        "Muy Satisfecho"))
tab <- as.data.frame(table(indicador))
```
```

Listing 1: *Data tab creada Rmarkdown*

Nuestro marco de datos es **tab** y con ello intente hacer gráficos como por ejemplo.

```
```{r graf, echo=FALSE,out.width="45%",fig.cap=" Grafico Likert",
           fig.pos="!htbp",fig.align='center'}

ggplot(tab,aes_(x="",y=tab$Freq,fill=factor(tab$indicador)))+
  geom_bar(width = 1,stat = "identity")+
  coord_polar("y")+
  geom_text(aes(label=paste(round(Freq/sum(Freq)*100,1),"%")),
            position=position_stack(vjust = 0.5))+
  theme_void(base_size = 18)+labs(fill="Indicador")

```
```

Figura 9.7: *ggplot en Rmarkdown*

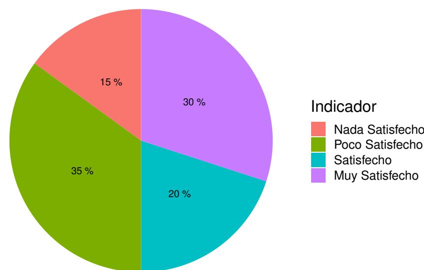


Figura 1: Grafico Likert

5.2.5 tablas

Las tablas creadas en markdown solo consiste en lineas verticales y horizontales, lo que hace muy sencillo la manipulación.

crear

Se intentará crear tablas manualmente, es importante agregar que también se puede agregar texto matemático y alineación por cada columna.

```
Tabla normal sin alinear

|   H1    | H2 | H3  | H4 |
|--------|----|-----|----|
|$\alpha$| 14 | 23  | 23 |
| 2      |    |     | 1  |

Table: Tabla normal

Tabla alineada por columnas

| izquierda |centro|derecha|
|:---------|:----:|------:|
| $\alpha$  | 14   | 23333 |
| **a**     | **b**|**_c_**|

Table: Tabla alineada
```

Tabla normal sin alinear

Cuadro 9.5: *Tabla normal*

H1	H2	H3	H4
α	14	23	23
2			1

Tabla alineada por columnas

Cuadro 9.6: *Tabla alineada*

H1	H2	H3
α	14	23333
a	**b**	***c***

librería

En la figura 9.6 agregar esta librería

```
library(kableExtra)
```

Una vez agregado necesitamos nuestra data **tab** (vea el Listing 1) y haga lo siguiente

````
```{r tabla,echo=F}
kable(tab,format="latex",booktabs=T,align="c",
                              caption="Tabla A")%>%
  kable_styling(latex_options ="hold_position")%>%
  column_spec(1,bold = T,color = "black")
```

Si quiere ver la referencia (vease **tabla \ref{tab:tabla}**)
````

Cuadro 3: Tabla A

indicador	Freq
Nada Satisfecho	3
Poco Satisfecho	17
Satisfecho	5
Muy Satisfecho	7

Si quiere ver la referencia (vease **tabla 3**)

6.2.6 bibliografía

Las referencias de documentos como:libros,artículos,revistas,enlaces y otros. Son almacenados desde un fichero externo llamado **bibliografia.bib**(no uses tilde), para saber como obtener referencias desde software visite la sección 3.5. Primero tenemos que agregar más elementos a nuestra cabecera:

Agregar a Cabecera

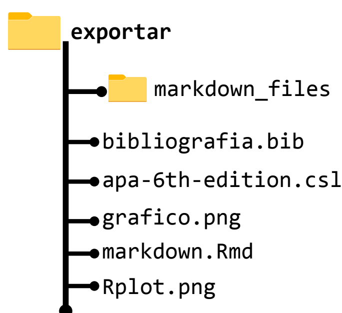
```
---

bibliography: bibliografia.bib
csl: apa-6th-edition.csl
output:
  pdf_document:
    citation_package: natbib
---
```

citation_package es el paquete con el cual se va a procesar la bibliografía, existen dos "natbib y biblatex"

El **csl** es el estilo que se le dará a nuestras referencias y zotero ofrece varios estilos, por favor ingrese en esta dirección **https://www.zotero.org/styles** o este otro **https://github.com/citation-style-language/styles** y descarga el que más se adapte a sus necesidades.

Por ejemplo tengo la siguiente bibliografía:



bibliografia.bib

```
@article{khan2020,
        title = {Level of satisfaction regarding ...},
        volume = {70},
        issn = {0030-9982},
        doi = {10.5455/JPMA.10977},
        pages = {452--456},
```

```
        number = {3},
        journaltitle = {{JPMA}. The Journal of ...},
        shortjournal = {J Pak Med Assoc},
        author = {Khan, Muhammad Lakhmir and ...},
        date = {2020-03},
        pmid = {32207424},
        keywords = {Level of satisfaction, ...}
}
...
...
```

Hago las siguientes referencias y en el último creo un encabezado "**`# Referencia`**" unicamente para imprimir la bibliografía

```

@khan2020 también incluyo el término...

El autor @khan2020 también ...

El autor [@khan2020] también ...

El autor [@khan2020, pp. 453-454] también ...

# Referencias

```

Compila el documento haciendo clic en el icono y elija pdf(**ver figura 9.5**) E intente usar natbib y biblatex en **citation_package**

```
---

output:
  pdf_document:
    citation_package: biblatex
---
```

Figura 9.8: *Natbib vs Biblatex*

(a) natbib

Khan et al. también incluyo el término. . .

El autor Khan et al. también . . .

El autor [Khan et al.] también . . .

El autor [Khan et al., pp. 453-454] también . .

Referencias

Muhammad Lakhmir Khan, Misbah Ghous, Imran Amjad, Munazza Nouman, and Irum Yaqoob. Level of satisfaction regarding physical therapy services in district poonch, azad jammu kashmir. 70(3):452–456. ISSN 0030-9982. doi: 10.5455/JPMA.10977.

(b) biblatex

Khan y col. [1] también incluyo el término. . .

El autor Khan y col. [1] también . . .

El autor [1] también . . .

El autor [1, pp. 453-454] también . . .

Referencias

[1] Muhammad Lakhmir Khan y col. "Level of satisfaction regarding physical therapy services in district Poonch, Azad Jammu Kashmir". En: *JPMA. The Journal of the Pakistan Medical Association* 70.3 (mar. de 2020), págs. 452-456. ISSN: 0030-9982. DOI: 10.5455/JPMA.10977.

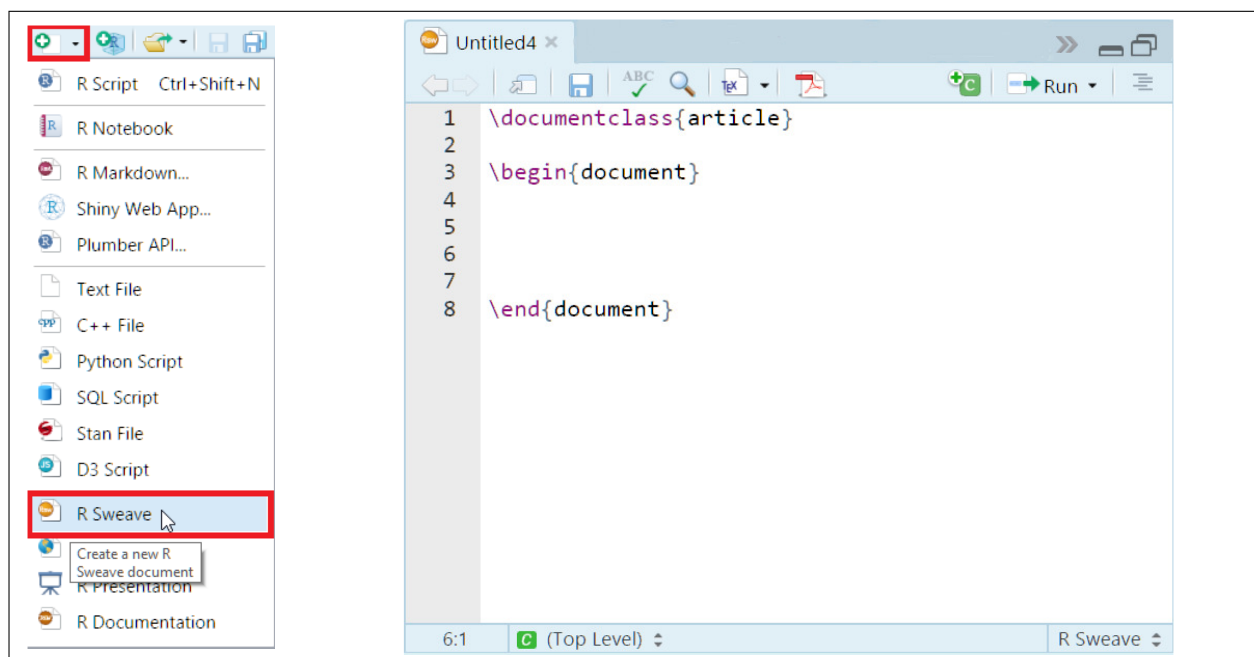
Descarga de Códigos



Puede usted visitar el repositorio github y encontrará todos los comandos escritos en este capitulo, recuerde visitar en este enlace **https://github.com/Moriand/Apuntes-R**

3.3 Latex

Hay IDEs del lenguaje R que tienen incorporado la función llamada **R Sweave** o mejor conocido en LaTeXcomo knitr. Si usted tiene Rstudio instalado, inicie y haga igual como se muestra en la imagen. *Para guardar la ventana de la derecha presione* ctrl + S



Latex es un sistema de composición avanzado de texto, su función es crear un documento de alta calidad como son: libros, artículos, reportes, cartas y otros. Knitr es una librería del lenguaje **R** que permite la intercomunicación entre Rscript y Latex, por medio de un fichero "**.Rnw**".

El inconveniente al usar Rstudio es que no compila las bibliografías por lo que se recurrirá a otros softwares libres como TeXmaker, TeXnicCenter y otros. Es importante dejar la bibliografía para el final caso que al compilar con Rstudio genera un fichero **.tex** y éste se lo compilará con los softwares mencionados anteriormente.

Para tratar de entender como es que funciona LaTeX, se presentará la estructura de un documento.

1 H1

1.1 H2

1.1.1 H3

1 H1

1.1 H2

1.1.1 H3

Configuración

```
\documentclass[11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[spanish]{babel}
\usepackage{amsmath}
\usepackage{graphicx}
\usepackage{microtype}
\usepackage[x11names,table]{xcolor}
\usepackage{hyperref}
\usepackage{geometry}
```

Cuerpo

```
\begin{document}

\section{H1}
\subsection{H2}
\subsubsection{H3}
%---- este es un comentario --
\section{H1}
\subsection{H2}
\subsubsection{H3}

\end{document}
```

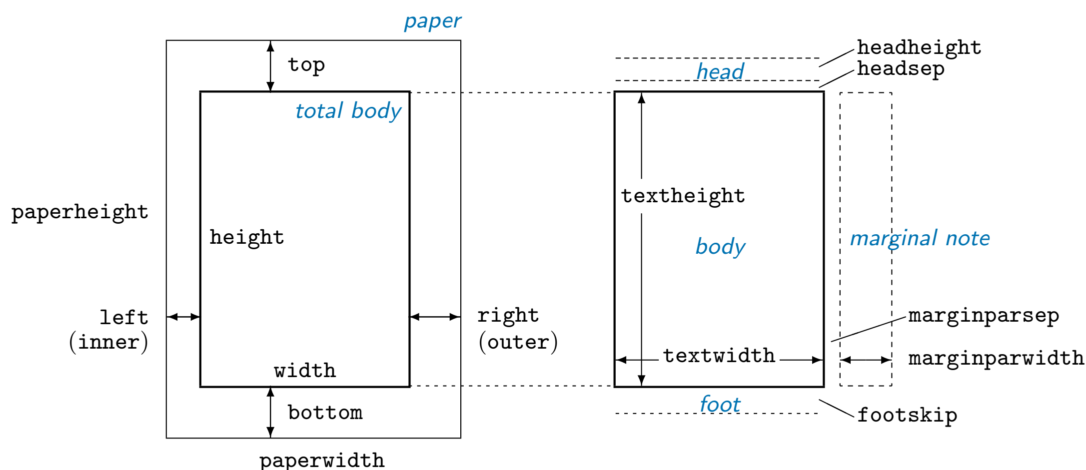
> **Importante:** Si por casualidad te genera error al momento de compilar, instala MiKTeX desde **https://miktex.org/download**

Se trabajará con la parte del **cuerpo** que es donde se comienza a redactar el documento. Muy bien que tal si empezamos con un comando que permite la entrada de código R.

```
<<  >>=


@
```

Figura 9.9: *Geometria Latex*



debajo del paquete **\usepackage{geometry}** agrege este código, recuerde que esto define la geometria de todo el documento.

```
\geometry{
  paper=a4paper,
  left=2.5cm,
  right=3.8cm,
  top=1.5cm,
  bottom=1.5cm,
  heightrounded,
  marginparwidth=51pt,
  marginparsep=17pt,
  headsep=24pt,
}
```

Si desea verificar la geometría actual de su documento, agregue este paquete después de geometry. Despues compile y observe

```
\usepackage{layout} % paquete

\makeatletter
\renewcommand*{\lay@value}[2]{%
  \strip@pt\dimexpr0.351459\dimexpr\csname#2\endcsname\relax\relax mm%
}
\makeatother

\begin{document}
\layout % crea una pagina con la geometria

\end{document}
```

Si desea elimine el paquete y el comando **\layout** para su normalidad.

1.3.1 Textos

Los textos en LaTeXson muy sencillos y es como escribir en un bloc de notas ya que no es posible observar como es el diseño o el texto que actualmente se esta escribiendo. A continuación se presentará el tamaño de letra:

Cuadro 9.7: *Tamaño de texto*

	Tipo de letra			
Escribir	**10pt**	**11pt**	**12pt**	**Produce**
{\tiny Tiny}	5pt	6pt	6pt	Tiny
{\scriptsize Script}	7pt	8pt	8pt	Script
{\footnotesize Foot}	8pt	9pt	10pt	Foot
{\small Small}	9pt	10pt	11pt	Small
{\normalsize Normal}	10pt	11pt	12pt	Normal
{\large large}	12pt	12pt	14pt	large
{\Large Large}	14pt	14pt	17pt	Large
{\LARGE Large}	17pt	17pt	20pt	Large
{\huge huge}	20pt	20pt	25pt	huge
{\Huge Huge}	25pt	25pt	25pt	Huge

Nota: Los 10,11 y 12pt se encuentra en la clase de documento `\documentclass [a4paper,12pt]{article}`
Importante: Las llaves sirven para indicar el límite del texto a convertir.

Hay algunos comandos que permiten hacer que el texto sea en negrita, itálica o cualquier otro.

Cuadro 9.8: *Estilo de texto*

Escribir	Produce
{\rm Roman }	Roman
{\em Enfático}	*Enfático*
\textbf {Negrita}	**Negrita**
\textit {Itálica}	*Itálica*
{\sf Sans Serif }	Sans Serif
\underline {Subrayado}	Subrayado

Párrafos

Los comandos que permiten crear párrafos son:

Cuadro 9.9: *Parrafos*

Escribir	Produce
`\paragraph {Titulo} ...`	**Titulo** ···
`\subparagraph {Titulo} ...`	**Titulo** ···

Entrada 1

```
\documentclass[11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath,amssymb}
     ⋮
\begin{document}
\section{H1}
\paragraph{Parrafo 1}
En cada viaje Mr. X sigue siendo un escritor ...
\subparagraph{Sub párrafo 1}
Uno de sus viajes Mr. X tuvo un problema ...
\paragraph{Parrafo 2}
Cada quien entendió a su manera, no obstante ...
\subparagraph{Sub párrafo 2}
Siempre decia que la felicidad ...
\end{document}
```

Genera 1

1 H1

Parrafo 1 En cada viaje Mr. X sigue siendo un escritor que esta muy ancioso de seguir escribiendo nuevas novelas que cuentan cada realidad subjetiva de un hombre quien dio la vida por un momento de felicidad ···

Sub párrafo 1 Uno de sus viajes Mr. X tuvo un problema con uno de los marines y es que en cuestión, él nunca prestaba atención a los ···

Parrafo 2 Cada quien entendió a su manera, no obstante él nunca se daba por vencido ni aunque derramara una gota de sangre en su boligrafo ···

Sub párrafo 2 Siempre decia que la felicidad es un tiempo muy corto que cada ser humano decidia si lo merecia ···

Alinear texto

Alinear un grupo de textos en LaTeXes muy sencillo y fácil, se usa los entornos flushleft, flushright y **center** "*unicamente solo cambia el texto que se encuentra dentro del entorno*
`\begin{} ..\end{}`".

Izquierda

Entrada 2

```
Texto arriba
\begin{flushleft}
..izquierda
\end{flushleft}
Texto abajo
```

Produce

Genera 2

Texto arriba

..izquierda

Texto abajo

Derecha

Entrada 3

```
Texto arriba
\begin{flushright}
..derecha
\end{flushright}
Texto abajo
```

Produce

Genera 3

Texto arriba

..derecha

Texto abajo

Centro

Entrada 4

```
Texto arriba
\begin{center}
..centrado
\end{center}
Texto abajo
```

Produce

Genera 4

Texto arriba

..centrado

Texto abajo

Espacios

Para dejar espacios en LaTeXtambién usamos comandos que son sencillos de memorizar, las medidas aceptadas son: puntos (**pt**), pulgadas (**in**), centímetros (**cm**), milímetros (**mm**), (**em**) y picas (**pc**)

Cuadro 9.10: *Espacios*

Escribir	Información
\hfill a	El texto **a** se mueve totalmente hacia la derecha
\vfill b	El texto **b** se mueve en su totalidad hacia abajo
\hspace {0.2in} c	El texto **c** es movido horizontalmente en **0.2in**
\vspace {0.2in} d	El texto **d** se mueve vertical(abajo) en **0.2in**
\par e	El texto **e** se escribe en una nueva linea

Listas

Las listas en LaTeXson de distintas formas: item (itemize),enumeracion (enumerate) y descriptivo (description). Cada una de ellas con sus respectivos diseños que tambien son editables los iconos o simbolos.

Para su edición algunos de los simbolos requieren agregar paquetes en LaTeX(vease la **figura 9.11**):

Figura 9.10: *Paquetes importantes*

```
\documentclass[11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath,array,amsfonts,latexsym,cancel,amssymb}
\usepackage{textcomp,wasysym} % <----
\let\Square\relax % importante
\usepackage{bbding} %  <-------------
...
```

Cuadro 9.11: *Símbolos Matemáticos para listas*

\triangle	$\triangle$	**\Box**	$\Box$	**\heartsuit**	$\heartsuit$
\rightarrowtail	$\rightarrowtail$	**\circledcirc**	$\circledcirc$	**\circledast**	$\circledast$
\boxtimes	$\boxtimes$	**\boxplus**	$\boxplus$	**\vartriangleright**	$\vartriangleright$
\wasytherefore	∴	**\circleddash**	$\circleddash$	**\multimap**	$\multimap$

Nota: si no esta dentro de un entorno matemático usar **$...$** (vea la sección 3.2)

Cuadro 9.12: *Símbolos no Matemáticos para listas*

\ding {47}	✏	\ding {52}	✔	\ding {56}	✘	\ding {60}	✜
\ding {54}	✖	\ding {45}	✍	\ding {64}	✠	\ding {70}	✦
\ding {72}	★	\ding {73}	☆	\ding {116}	▼	\ding {109}	❍
\ding {220}	➜	\ding {226}	➢	\ding {228}	➤	\ding {229}	➥

Nota: no requieren de **$** para usarlo como texto normal.

Modo normal

Entrada 5

```
\textbf{listas forma item}
\begin{itemize}
    \item a y b
    \begin{itemize}
        \item sub a y b
    \end{itemize}
    \item b y c
\end{itemize}
\textbf{listas número}
\begin{enumerate}
    \item a y b
    \begin{enumerate}
        \item sub a y b
        \item sub c y d
    \end{enumerate}
    \item b y c
\end{enumerate}
\textbf{listas descriptivo}
\begin{description}
\item[Flor] Parte de planta
\item[Ave] Animal vertebrado
\end{description}
```

Genera 5

listas forma item

- a y b
    - sub a y b
- b y c

listas número

1. a y b
    a) sub a y b
    b) sub c y d
2. b y c

listas descriptivo

Flor Parte de planta

Ave Animal vertebrado

Cambiando símbolos, si quieres usar **[label=]** debes primero agregar este paquete
\usepackage{enumitem} donde se encuetra tus paquetes(ver figura 9.11)

Entrada 6

```
\textbf{listas forma item}
\begin{itemize}
    \item[$\checkmark$] a y b
    \item[$\bigodot$] b y c
\end{itemize}

usando \textbf{label}

\begin{itemize}[label=$\bigstar$]
    \item a y b
    \begin{enumerate}
        \item uno
        \item dos
    \end{enumerate}
    \item c y d
```

Genera 6

listas forma item

- ✔ a y b
- $\bigodot$ b y c

usando **label**

- ★ a y b
  1. uno
  2. dos
- ★ c y d

2.3.2 Ecuaciones

Figura 9.11: *Paquetes importantes*

```
\documentclass[11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath,array,amsfonts,latexsym,cancel,amssymb}
...
```

Lo enriquecedor y maravilloso de LaTeXes la forma como imprime las ecuaciones, a continuación en la siguiente tabla se muestra algunos símbolos más usados en ciencias matemáticas

Cuadro 9.13: *Algunos símbolos matemáticos*

\alpha	α	**\epsilon**	ϵ	**\leq**	$\leq$	**\cong**	$\cong$	**\pi**	π
\beta	β	**\theta**	θ	**\geq**	$\geq$	**\neq**	$\neq$	**\phi**	ϕ
\gamma	γ	**\lambda**	λ	**\in**	$\in$	**\&**	&	**\delta**	δ
\delta	δ	**\mu**	μ	**\not \in**	$\notin$	**\$**	$	**\Omega**	Ω
\sigma	σ	**\pm**	$\pm$	**\not** =	$\neq$	**\%**	%	**\therefore**	$\therefore$

Nota: Para escribir **un símbolo** como texto use el dolar, ejemplo **α**

Cuadro 9.14: *Funciones y Operadores*

\sigma ^{2}	σ^2	**\ln** _{b}{a}	$\ln_b a$
32_{2}	32_2	**\dfrac** {b}{a}	$\dfrac{b}{a}$
\sum _{b}^{a} n_{i}	$\sum_b^a n_i$	**\tfrac** {b}{a}	$\tfrac{b}{a}$
\sqrt [n]{12}	$\sqrt[n]{12}$	**\lim** _{a **\rightarrow** b}	$\lim_{a\rightarrow b}$
\int _{b}^{a} 2x	$\int_b^a 2x$	**\iint** _{b}^{a} 2x	$\iint_b^a 2x$
\prod _b^a	$\prod_b^a$		

Cuadro 9.15: *Sombreros Matemáticos*

`\dot {a}`	$\dot{a}$	`\check {c}`	$\check{c}$	`\tilde {d}`	$\tilde{d}$	`\vec {d}`	$\vec{d}$
`\ddot {b}`	$\ddot{b}$	`\grave {s}`	$\grave{s}$	`\breve {c}`	$\breve{c}$	`\widehat {ab}`	$\widehat{ab}$
`\hat {\beta }`	$\hat{\beta}$	`\acute {d}`	$\acute{d}$	`\bar {d}`	$\bar{d}$	`\widetilde {ab}`	$\widetilde{ab}$

Nota: Para mejorar el estilo, usar `$\displaystyle \dot{a} $`.
El espacio en modo matemático es `$\hat{a}\;\;\bar{b}$` y genera $\hat{a}\;\;\bar{b}$

Entrada 7

```
Por ejemplo $\alpha$ y el $\beta$
\[

 \bar{x}=\dfrac{\sum_{i=1}^{n}x_i}{n}

\]
también se usa
$$\sum_{b}^{a} n_{i}$$
mejora el diseño con displaystyle
$\displaystyle\lim_{a \rightarrow b}$
```

Genera 7

Por ejemplo α y el β

$$\bar{x}=\dfrac{\sum_{i=1}^{n}x_i}{n}$$

también se usa

$$\sum_{b}^{a} n_{i}$$

mejora el diseño con **displaystyle** $\displaystyle\lim_{a \rightarrow b}$

Ecuaciones alineadas cortas

Usamos el entorno de **align** para alinear ecuaciones matemáticas. Cuando usa **`\begin{align*}...\end{align*}`** con asterisco estará omitiendo referencias. Para usar el texto en negrita usar **`\pmb{}`**

Entrada 8

```
He aquí un texto que bien pudo ser un párrafo ...
\begin{align*}
  \pmb{\hat{\beta}} &= \dfrac{\pmb{S_{xy}}} {\pmb{S_{xx}}} \\
  \pmb{\int_{0}^2} x^{2}dx &= \dfrac{8}{3}
\end{align*}
Recuerda que el texto matemático esta en el centro
```

Genera 8

He aquí un texto que bien pudo ser un párrafo ...

$$\begin{aligned} \pmb{\hat{\beta}} &= \dfrac{\pmb{S_{xy}}}{\pmb{S_{xx}}} \\ \pmb{\int_{0}^2} x^{2}dx &= \dfrac{8}{3} \end{aligned}$$

Recuerda que el texto matemático esta en el centro

También se encuentra el entorno **\begin{equation}...\end{equation}**, mayormente los uso para ecuaciones fundamentales por el motivo que no acepta **&** para su alineación, ademas la ecuación a escribir tiene que ser de una sola linea.

Cuando intentemos usar **\begin{align}...\end{align}** con asteriscos, significa que no hay referencias y no imprimirá el número de la ecuación

Entrada 9

```
primera ecuación con referencia
\begin{equation}
S_{xx} = \sum (x_{i}-\bar{x})^2
\label{eq:sx} %referencia
\end{equation}
Segunda ecuación,por favor no deje líneas vacías
\begin{equation}
% es preferible comentar
S_{xy} = \sum y_{i}(x_{i} -\bar{x})
% que a dejarlo en blanco
\label{eq:sy}
\end{equation}
Ahora en otro entorno
\begin{align}
  \pmb{\hat{\beta}} &= \dfrac{\pmb{S_{xy}}}{\pmb{S_{xx}}}
\tag{véase \eqref{eq:sy} y este \eqref{eq:sx} }\\
\label{eq:lineal}
\end{align}
Para referencias use \ref{eq:sy} al igual que en el otro \ref{eq:sx}.
Pero también tenemos la ecuación general \ref{eq:lineal}
```

Genera 9

primera ecuación con referencia

$$S_{xx} = \sum (x_i - \bar{x})^2$$

Segunda ecuación,por favor no deje líneas vacías

$$S_{xy} = \sum y_i(x_i - \bar{x})$$

Ahora en otro entorno

$$\boldsymbol{\hat{\beta}} = \frac{\boldsymbol{S_{xy}}}{\boldsymbol{S_{xx}}} \tag{véase (3.2)}$$

(9.1)

Para referencias use 3.2 al igual que en el otro 3.2. Pero también tenemos la ecuación general 9.1

Ecuaciones alineadas grandes

Cuando estas apunto de escribir expresiones enormes que puede caber en varias lineas, usas lo siguiente:

> **Importante:** Por favor trate de no dejar lineas vacías(en blanco). Intente compilar dos veces para que en referencias no aparezca el símbolo **?**

Entrada 10

```
\begin{multline}
\pmb{f(x)} = ax^{10}+32+ax^{9}+12x+23+ax^{8}+11x+8+ax^{7}+
              9x+4+ax^{6}+8x+9+ax^{5}+8x \\
+12+ax^{4}+7x+7+ax^{3}+45x+3+ax^{2}+5x+3+ax
\label{eq:enorme}
\end{multline}
Referencia con \textbf{\ref{eq:enorme}}
```

Genera 10

$$\boldsymbol{f(x)} = ax^{10}+32+ax^{9}+12x+23+ax^{8}+11x+8+ax^{7}+9x+4+ax^{6}+8x+9+ax^{5}+8x \\ +12+ax^{4}+7x+7+ax^{3}+45x+3+ax^{2}+5x+3+ax \quad (9.2)$$

Referencia con **9.2**

El paquete que permite colorear "`\usepackage [x11names,table]{xcolor}`" y este otro "`\usepackage {cancel}`" sirve para tachar expresiones.

Cuadro 9.16: *Tachado de expresiones y coloreado*

Escribir	Produce
`\cancel {a}`	$\cancel{a}$
`\bcancel {2x^{0}}`	$\bcancel{2x^{0}}$
`\xcancel {0x}`	$\xcancel{0x}$
`\cancelto {\infty }{3_{i}}`	$\cancelto{0}{3_{i}}$
`\textcolor {red}{\int {2}}`	$\int 2$
`{\color {blue}3x^{2\alpha } }`	$3x^{2\alpha}$

Nota 1: Cancel funciona en entorno matemático(**align,multline,equation**), caso contrario usarlo entre **$**, por ejemplo `$\cancel{a}$`.
Nota 2: El coloreado de texto es usado en cualquier lugar, no se necesita de entorno matemático.

En align se usa **&** para determinar su alineación y estas lineas \\ para escribir en el siguiente renglón (**En el ultimo no usarlo**)

```
begin{align}
a &= b    &    a &= b   &   a &= b    &   a &= b \\
a &= b    &    a &= b   &   a &= b    &   a &= b  \\
a &= b    &    a &= b   &   a &= b    &   a &= b
end{align}
```

Entrada 11

```
\begin{align*}
 \cancel{0}+2x &= \cancelto{1}{a} & \xcancel{0b} &= 3x \\
        0  &=\textcolor{blue}{2x} &  0 &= \cancel{\frac{3x}{0}} \\
 \bcancel{0} &= 2x          &  0 &= 0
\end{align*}
```

Genera 11

$$\begin{aligned} \cancel{0}+2x &= \cancelto{1}{a} & \xcancel{0b} &= 3x \\ 0 &= 2x & 0 &= \cancel{\frac{3x}{0}} \\ \bcancel{0} &= 2x & 0 &= 0 \end{aligned}$$

Si quiere hacer pequeños espacios en modo matemático use "\;", si duplica, triplica o quintuplica el espacio aumentará:
\;\;\;\;\;

```
$$
 x= 2\;\;\;\;\;x+b
$$
```

$$x = 2\;\;\;\;\;x+b$$

Mas información

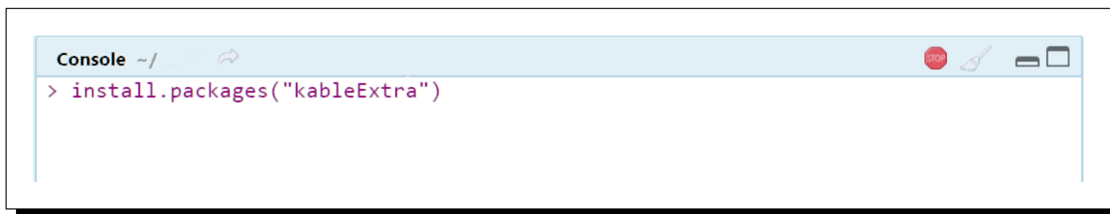
Para las ecuaciones revise este libro en el "capitulo 4" [1] y también este otro [10]. Ambos libros están disponibles en la web y están muy bien detallados.

3.3.3 Tablas

Mediante librerías

Hay librerías en **R** que permiten generar tablas como es el caso de **kableExtra** y **xtable**. Desde la consola de Rstudio o cualquier otro IDE, ejecute el siguiente código:

```
> install.packages("kableExtra")
> install.packages("xtable")
```





Recuerda que para escribir código R tienes que obligatoriamente usar estos símbolos << >>= y al final @

```
<<label,echo=F,results='asis' >>=
# aquí escribir todo lo relacionado con R
@
```

Cuadro 9.17: *Argumentos en Tabla*

Argumento	función
label	es la etiqueta que se tomará como referencia al usar `\ref {tab:label}`, importante "**tab:**"
echo	(booleano) si es **TRUE** imprime el código escrito.
background	(char) es usado cuando `echo=TRUE`, usar color en hexadecimal como `background = '##ecf5fc'`
eval	(booleano) si es **TRUE** evalua la linea que tiene el código
results	(char) si usas **xtable** recomiendo usar como argumento `'asis'`

Nota: **TRUE** es lo mismo que escribir una **T** mayúscula.

Cuidado: no confundas el símbolo para comentar, **en el lenguaje R se usa #** y para latex es %.

Una vez instalado, dejamos la consola y nos dedicamos al documento **.Rnw** y escribimos lo siguiente como ejemplo:

```
\documentclass[11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[spanish]{babel}
\usepackage{amsmath}
\usepackage{graphicx,wrapfig}
\usepackage{longtable,booktabs,multirow}
\usepackage{array,float,pdflscape,tabu}
\usepackage{threeparttable,threeparttablex}
\usepackage[normalem]{ulem}
\usepackage{xcolor,colortbl,makecell}
%==========================================
<<echo=FALSE,warning=FALSE,message=FALSE>>=
library(knitr)
library(kableExtra)
library(xtable)
options(xtable.comment = FALSE)   1
@
%------------------------------------------
<<echo=FALSE>>=
```

```
#----------          2
large <- function(x){
  paste0('{\\Large{\\bfseries ',x,'}}')
}
#---------------------------------------
nombre <- c("c1","c2","c3","c4","c5","c6")
tb = data.frame(
  "x"= sample(1:15,size = 6,replace=T),
  "y"= runif(n=6, min = 8, max = 26 ),
  "z"= sample(30:65,size = 6,replace = T)
   )
rownames(tb) <- nombre
#-------------------
media <- round(mean(tb$x),2)
@
%===============================================
\begin{document}
\section{Reporte}
Intente crear una tabla desde \textbf{R}

<<tabla,echo=F>>=
kable(tb,format="latex",booktabs=T,align="c",caption="Tabla A",
                                                          row.names=1) %>%
  kable_styling(latex_options ="hold_position") %>%
  column_spec(1,bold = T,color = "black")
@
Se puede referenciar desde este comando: {\bf Cuadro \ref{tab:tabla} }
\begin{align*}
\bar{x} &=\dfrac{\sum_{i=1}^{n}x_i}{n} \\
\bar{x} &= \Sexpr{media}                   3
\end{align*}

Intentar con la libreria \textbf{xtable} que es mas
configurable
<<echo=F,results='asis'>>=
tablas <- xtable(x = tb,caption = "tabla loca")
print(tablas,sanitize.colnames.function = large )   4
@

<<echo=T,eval=T,background='#ecf5fc' >>=
ce= c(12,2,3)
print(2*ce)
@

\end{document}
```

Una vez escrito todo dale clic al icono del pdf para compilar:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[spanish]{babel}
\usepackage{amsmath}
\usepackage{graphicx,wrapfig}
\usepackage{longtable,booktabs,multirow}
\usepackage{array,float,pdflscape,tabu}
\usepackage{threeparttable,threeparttablex}
\usepackage[normalem]{ulem}
\usepackage{xcolor,colortbl,makecell}
%==========================================
<<echo=FALSE,warning=FALSE,message=FALSE>>=
library(knitr)
library(kableExtra)
library(xtable)
options(xtable.comment = FALSE)
@
%-----------------------------------------
<<echo=FALSE>>=
#----------
large <- function(x){
  paste0('{\\Large{\\bfseries ',x,'}}')
```

El resultado es:

1. Reporte

Previa

Intente crear una tabla desde **R**

Cuadro 1: Tabla A

	x	y	z
c1	4	22.13087	36
c2	10	17.22997	50
c3	7	17.39483	64
c4	6	20.10082	65
c5	9	15.00441	33
c6	7	22.05967	61

Se puede referenciar desde este comando: **Cuadro 1**

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\bar{x} = 7{,}17$$

Intentar con la libreria **xtable** que es mas configurable

	x	**y**	**z**
c1	4	22.13	36
c2	10	17.23	50
c3	7	17.39	64
c4	6	20.10	65
c5	9	15.00	33
c6	7	22.06	61

Cuadro 2: tabla loca

```
ce= c(12,2,3)
print(2*ce)

## [1] 24  4  6
```

1

Manualmente

Se iniciará primero por crear tablas simples y para ellos es necesario incluir estos paquetes: **\usepackage{multicol,multirow,booktabs}** en la figura 9.11:

Agregar paquetes

```
\documentclass[11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[spanish]{babel}
\usepackage{amsmath,array,amsfonts,latexsym,cancel,amssymb}
%-----------------> Listas  <-------
\usepackage{textcomp,wasysym}
\let\Square\relax
\usepackage{bbding}
\usepackage{enumitem}
%-----------------> Tablas <--------
\usepackage[x11names,table]{xcolor}
\usepackage[capposition=top]{floatrow} % título arriba
\usepackage{multicol,multirow,booktabs}
\usepackage{threeparttable}
\usepackage{longtable}
```

Por ejemplo la estructura es la siguiente

```
\begin{table}[!htb]
    \centering
    \begin{tabular}{|c|c|c|c|}
    \hline
        &  &  & \\ \hline
        &  &  & \\ \hline
        &  &  &  \\ \hline
        &  &  &  \\ \hline
        &  &  &  \\ \hline
    \end{tabular}
    \caption{Título}
    \label{tab:my_lab}
\end{table}
```

Cuadro 1: Título

El entorno principal es **\begin{table}[!htb]...\end{table}** únicamente sirve para referencia en la tabla de contenidos dedicado solo a tablas, para mostrar solo debe escribir debajo de **begin{document}** el comando **\listoftables** y aparecerá el índice delas tablas en orden descendente.

Despues se encuentra otro comando **\centering**, indicando que centrará la tabla.

el entorno **\begin{tabular}{ }** inicia el cuadro y lo importante es lo que se encuentra entre las llaves:

"**{|c|}**", éste símbolo "**|**" crea lineas verticales en columnas. La letra "**c**" significa la alineación del contenido y que también es en columnas, preste atención a la siguiente tabla 9.18:

Cuadro 9.18: *Alineación por columnas en tablas*

carácter	significado
l	alineación hacia la izquierda(**l**eft)
c	alineación hacia el centro
r	alineación hacia la derecha(**r**igth)
m{8cm}	alinea al medio y limita el ancho de toda la columna a 8cm
p{8cm}	alinea tipo párrafo y limita el ancho de toda la columna a 8cm

Nota 1: para **m** y **p** también puede aplicar otras medidas(vea la **sección 3.1**)
Nota 2: si tiene 3 columnas,debes tener **{ccc}** tres caracteres de alineación.

El "**\hline**" crea una linea horizontal por filas.
El "**&**" sirve para separar celdas.
las dos barras invertidas "****" significa salto de linea
El "**\caption{}**" se escribe el título de la tabla
El "**\label{tab:}**" es la etiqueta que sirve para hacer referencia al usar **\ref{tab:}**

Pasaremos al coloreado de las filas y celdas dentro de una tabla.
Preste atención al siguiente ejemplo:

Entrada 12

```
\begin{table}[!htb]
    \centering
    \rowcolors{3}{DeepSkyBlue1}{Coral1} %fila 3 y 4
    \begin{tabular}{|c|c|c|c|}
    \hline
      \rowcolor{gray!30} &   &      T1        & \\ \hline
                    2    &  &\cellcolor{green} & \\ \hline
                    3    &  &       c          &  \\ \hline
                    4    &  &       d          &  \\ \hline
    \end{tabular}
    \caption{Título 2}
    \label{tab:my_lab2}
```

```
\end{table}
```

Genera 12

Cuadro 2: Título 2

		T1	
2			
3		c	
4		d	

\rowcolor{gray!30}
\cellcolor{green}
\rowcolors{3}{DeepSkyBlue1}{Coral1}

También se puede pintar por columnas pero requiere crear nuevos comandos como se muestra a continuación:

```
\newcolumntype{nombre}{>{}alineación}
```

Un comando para pintar el texto negrita a blanco:

```
\newcommand{\nombre}[1]{\textcolor{color}{\textbf{#1}} }
```

Entrada 13

```
...
%--------> crear comandos <-----
\newcolumntype{a}{>{\columncolor{Brown1}}c}
\newcolumntype{e}{>{\columncolor{DarkOrange1}}c}
%---------------
\newcommand{\tex}[1]{\textcolor{white}{\textbf{#1}}}
%-------------------------------

\begin{document}
....
....
\begin{table}[!htb]
    \centering
    \begin{tabular}{|a|c|e|}
    \hline
      \tex{T1} & T2 & \tex{T3} \\ \hline
               &    &          \\
               &    &          \\
               &    &          \\ \hline
    \end{tabular}
    \caption{Título 3}
```

```
    \label{tab:my_lab3}
\end{table}

\end{document}
```

Genera 13

Cuadro 3: Título 3

T1	T2	T3

para mas información de los colores ingresa en esta url
http://ctan.dcc.uchile.cl/macros/latex/contrib/xcolor/xcolor.pdf

Si te parece que no es sencillo crear tablas, en la comunidad de LaTeXhan creado una extensión que es compatible con Microsoft Excel, llamada “**Excel2LaTeX**” y lo puedes descargar desde la siguiente dirección:
https://github.com/krlmlr/Excel2LaTeX/releases

Esta extensión es compatible con todas las versiones de Excel, así que aquí tienes otra alternativa para crear tablas.

Las tablas anteriores son con estilo, pero que tal si se requiere de tablas tipo reportes o tesis. Abajo te presento en la tabla 9.19 las lineas que son usadas :

Cuadro 9.19: *Lineas en tablas*

Lineas	Dibuja
`\toprule`	linea superior
`\midrule`	linea central
`\cmidrule (lr){i-j}`	pequeñas lineas
`\bottomrule`	linea final

El comando `\cmidrule(lr){i-j}` es:

lr recorta la linea de izquierda(**l**eft) a derecha (**r**ight)

i-j representa las celdas de una fila, por ejemplo 2-4

Cuadro 9.20: *Unir filas y columnas en tablas*

Comando	Descripción
`multirow{m}[vspace]{*}{texto}`	unir celdas por filas(a la derecha)
`multicolumn{n}{lcr}{texto}`	unir celdas por columna(hacia abajo)

Nota 1: **m** y **n** son números de celdas.
Nota 2: Los coloreados de filas y columnas no funcionan.

Ejemplo 1

Entrada 14

```
\begin{table}[!htbp]
  \centering
  \begin{tabular}{ccc}
    \toprule
   \bf  Estado &\bf Label &\bf Valor \\ \midrule
      Mal     & ing        & 12  \\
      Regular & salt       & 4   \\
      Bueno   & run        & 6  \\ \midrule
     \multicolumn{2}{c}{\bf Total } & 22 \\ \bottomrule
    \end{tabular}
    \caption{Caption}
    \label{tab:my_lab4}
\end{table}
```

Genera 14

Cuadro 4: Caption

Estado	Label	Valor
Mal	ing	12
Regular	salt	4
Bueno	run	6
Total		22

Intente observar esta imagen

Sexo	Educación		
	Primaria	Secundaria	Superior
Hombre	1	0	0
Mujer	0	1	0

En a se ha usado "\multicolumn" une tres columnas.

Para el caso de b es "\multirow" por tratarse de la union de dos filas de la primera columna.

Entrada 15

```
\begin{table}[!htbp]
  \centering
 \begin{tabular}{cccc}
   \toprule
   \multirow{2}[7]{*}{Sexo}  &
   \multicolumn{3}{c}{  Educación  } \\ \cmidrule{2-4} %
      & Primaria & Secundaria  & Superior \\ \cmidrule(r){1-1}
                                            \cmidrule(lr){2-2}
                                            \cmidrule(lr){3-3}
                                            \cmidrule(l){4-4}
         Hombre  & 1 & 0  & 0   \\
         Mujer & 0 & 1  & 0   \\ \bottomrule
    \end{tabular}
    \caption{Caption}
    \label{tab:my_lab5}
```

```
\end{table}
```

Genera 15

Cuadro 5: Caption

	Educación		
Sexo	Primaria	Secundaria	Superior
Hombre	1	0	0
Mujer	0	1	0

El paquete "**threeparttable**" permite añadir notas debajo de la tabla:

Entrada 16

```
\begin{table}[!htbp]
    \centering
    \begin{threeparttable}
    \begin{tabular}{ccc}
        \toprule
      \bf  Estado  &\bf Label  &\bf Valor \\ \midrule
        Mal      & ing  & 12  \\
        Regular & salt  & 4   \\
        Bueno   & run  & 6  \\ \midrule
        \multicolumn{2}{c}{\bf Total } & 22 \\ \bottomrule
    \end{tabular}
    \begin{tablenotes} %-----
        \item \textbf{Nota: } la tinta se agota
    \end{tablenotes} %-------
    \end{threeparttable}
    \caption{Caption}
    \label{tab:my_lab6}
\end{table}
```

Genera 16

Cuadro 6: Caption

Estado	Label	Valor
Mal	ing	12
Regular	salt	4
Bueno	run	6
Total		22

Nota: la tinta se agota

Algunos comandos como “**\resizebox**” permite auto ajustar el ancho de la tabla, afectando el tamaño de la letra, se vuelve importante cuando tienes una tabla demasiada ancha:

```
\begin{table}[]
...
\resizebox{\textwidth}{!}{%
\begin{tabular}{ccc}
...
\end{tabular}
}%
...
```

Este otro entorno “**\begin{longtable}**” que es para la creación de enormes tablas, como por ejemplo:

Entrada 17

```
\begin{center} %----------> centrar tabla
\begin{longtable}{cccc}
\caption{Título de tabla} \\
\toprule
%==========  inicia cabecera
\multirow{2}[7]{*}{Sexo} &
   \multicolumn{3}{c}{  Educación } \\ \cmidrule{2-4} %
      & Primaria & Secundaria  & Superior \\ \cmidrule(r){1-1}
                                            \cmidrule(lr){2-2}
                                            \cmidrule(lr){3-3}
                                             \cmidrule(l){4-4}
\endhead
%========== fin cabecera
```

```
\hline \multicolumn{4}{r}{\textit{Continua en la
                                        siguiente pagina}} \\
\endfoot
%=========  pie de pagina
\bottomrule
\endlastfoot
\label{tab:my_tab7} %---> referencia
%========= fin pie de pagina
H & 2 & 3 & 4\\ H & 1 & 3 & 4\\ H & 1 & 3 & 4\\ H & 2 & 3 & 4\\
H & 7 & 3 & 7\\ H & 1 & 3 & 4\\ H & 2 & 3 & 4\\ M & 2 & 1 & 4\\
H & 2 & 7 & 6\\ M & 4 & 3 & 4\\ H & 4 & 3 & 4\\ M & 2 & 3 & 1\\
M & 2 & 3 & 1\\ M & 1 & 3 & 4\\ H & 2 & 3 & 4\\ M & 2 & 3 & 6\\
M & 2 & 3 & 4\\ M & 2 & 3 & 1\\ H & 4 & 3 & 7\\ M & 2 & 1 & 4\\
H & 6 & 3 & 6\\ M & 4 & 3 & 4\\ M & 2 & 3 & 4\\ M & 4 & 3 & 4\\
M & 2 & 3 & 1\\ H & 1 & 3 & 6\\ M & 2 & 7 & 4\\ M & 2 & 3 & 4\\
H & 2 & 3 & 4\\ H & 2 & 3 & 1\\ M & 2 & 3 & 4\\ M & 2 & 3 & 1\\
M & 6 & 7 & 1\\ M & 4 & 3 & 4\\ M & 2 & 1 & 4\\ M & 2 & 3 & 6\\
H & 2 & 3 & 6\\ M & 2 & 7 & 4\\ M & 2 & 3 & 1\\ M & 2 & 3 & 4\\
H & 6 & 3 & 1\\ M & 2 & 3 & 4\\ M & 2 & 3 & 4\\ M & 2 & 3 & 1\\
H & 2 & 3 & 4\\ M & 2 & 3 & 4\\ M & 2 & 3 & 4\\ M & 2 & 3 & 4\\
\end{longtable}
\end{center}

la tabla \ref{tab:my_tab7} % --> referencia
```

Genera 17

Cuadro 11: Título de tabla

Sexo	Educación		
	Primaria	Secundaria	Superior
H	2	3	4
H	1	3	4
H	1	3	4
H	2	3	4
H	7	3	7
H	1	3	4
H	2	3	4
M	2	1	4
H	2	7	6
M	4	3	4
H	4	3	4
M	2	3	1
M	2	3	1
M	1	3	4
H	2	3	4
M	2	3	6
M	2	3	4
M	2	3	1
H	4	3	7
M	2	1	4
H	6	3	6
M	4	3	4
M	2	3	4
M	4	3	4

Continua en la siguiente pagina

Cuadro 11: Título de tabla

Sexo	Educación		
	Primaria	Secundaria	Superior
M	2	3	1
H	1	3	6
M	2	7	4
M	2	3	4
H	2	3	4
H	2	3	1
M	2	3	4
M	2	3	1
M	6	7	1
M	4	3	4
M	2	1	4
M	2	3	6
H	2	3	6
M	2	7	4
M	2	3	1
M	2	3	4
H	6	3	1
M	2	3	4
M	2	3	4
M	2	3	1
H	2	3	4
M	2	3	4
M	2	3	4
M	2	3	4

de acuerdo a la tabla 11 referenciado

4.3.4 Figuras

Hay librerías como **ggplot2,plotrix,RGL y plotly(shiny)** que generan un perfecto diseño en R y lo mejor de todo es que se puede exportar a pdf sin perder calidad alguna. Dentro de estos símbolos se discutirá la dimensiones de los gráficos << >>=

```
> install.packages("ggplot2")
> install.packages("rgl")
> install.packages("plot3D")
#------------------------
> install.packages("plotrix")
```

Es recomendable tener la versión de **R** actualizado

Cuadro 9.21: *Argumentos en Figuras*

Argumento	función
label	Es la etiqueta que sirve para referenciar en latex por ejemplo: `\ref {fig:label}`, importante "**fig:**"
fig.pos	(char) Posición de la figura: arriba(t), flotante(h),al final(b). Por ejemplo: `fig.pos='!htbp'`
fig.height	(numérico) Es el alto de la figura (acepta valores de 1 hasta 7).Por ejemplo: `fig.height=4`
fig.width	(numérico) Es el ancho de la figura (acepta valores de 1 hasta 7). Por ejemplo: `fig.width=7`
out.width	(char) Se reduce o aumenta la imagen en forma proporcional (de 1 hasta 100 %), por ejemplo: `out.width='40%'`
out.extra	(char) Se trata de agregar argumentos extras que son compatibles con latex, por ejemplo rotar una imagen: `out.extra='angle=45'`
fig.cap	(char) Aquí ingrese el título de la figura, por ejemplo: `fig.cap='Figura N°1..'`
fig.ncol	(numérico) Para usarlo necesitas el paquete `\usepackage{subfig}` en LaTeX. Este argumento indica el numero de columnas.
fig.subcap	(char) Son los sub nombres a cada figura y si usaste el **fig.ncol**, hacer que coincidan en **n-subnombres** `fig.subcap=c('f1','f2',..n)`

Al usar **fig.ncol** se debe reducir cada figura, por ejemplo si **fig.ncol=2** que crea dos columnas y **n-filas**, usar `out.width="0.48\\textwidth"`. También debes saber que todos los argumentos en << >>= **se escriben en una sola linea.**

```
......
\usepackage{subfig} %<----------
......
```

```
%===========================================
«echo=FALSE,warning=FALSE,message=FALSE»=
library(knitr)
.....
library(ggplot2)
library(rgl)
library(plot3D)
library(plotrix)
@
%----------------------------------------
```

Usando estos valores aleatorios como nuestra fuente de datos

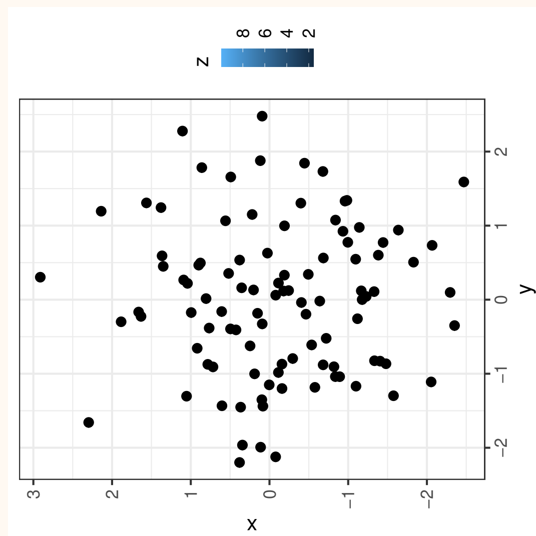
```
«echo=FALSE»=
tb = data.frame(
  "x"= sort(rnorm(100)),
  "y"= rnorm(100),
  "z"= runif(n=100, min = 1, max = 10)
   )
#----------------------------------------
sp <- sample(1:4,size = 20,replace = T)
x3 <- factor(x = sp ,levels = c(1:4),
               labels = c("Nada Satisfecho",
                          "Poco Satisfecho",
                          "Satisfecho",
                          "Muy Satisfecho"), ordered = T )
#--------------------
lk <- table(x3)
tab <- as.data.frame(lk)
@
%==============================================
```

Por ejemplo en **ggplot** una gráfica de puntos con un angulo de 90:

Entrada 18

```
«figura1,echo=F,out.width='60%',out.extra='angle=90'»=
grafico1 <- ggplot(data = tb, aes(x=y,y=x,fill=z))+
  geom_point(size=4)+
  theme_bw(base_size = 20)
#
grafico1
@
```

Genera 18

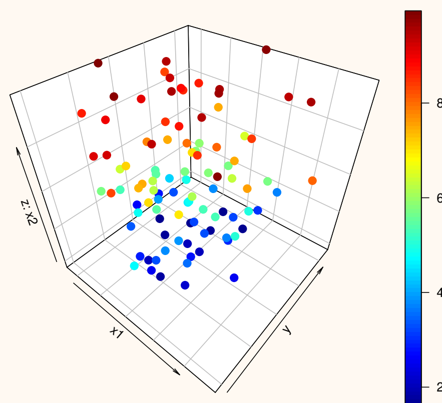


Por ejemplo en **plot3D(rgl)** grafico de dispersión

Entrada 19

```
«figura2,echo=F,out.width='60%'»=
scatter3D(x=tb$x,y=tb$y,z=tb$z,cex=1.5,bty="b2",xlab="x1",
          ylab="y",zlab="z: x2",pch=16,col.panel="#99A69B")
@
```

Genera 19



Usando nuevamente **ggplot**, un gráfico pie y para referencias usa `\ref{fig:fig3}`, no te olvides de "`echo=FALSE`"

Entrada 20

```
«fig3,out.width='70%',fig.cap='Grafico Likert',fig.pos='!hbt'»=
grafico3 <- ggplot(tab,aes_(x="",y=tab$Freq,fill=factor(tab$x3)))+
  geom_bar(width = 1,stat = "identity")+
  coord_polar("y")+
  geom_text(aes(label=paste(round(Freq/sum(Freq)*100,1),"%")),
            position=position_stack(vjust = 0.5))+
  theme_void(base_size = 18)+labs(fill="Indicador")
#
grafico3
@
```

Genera 20

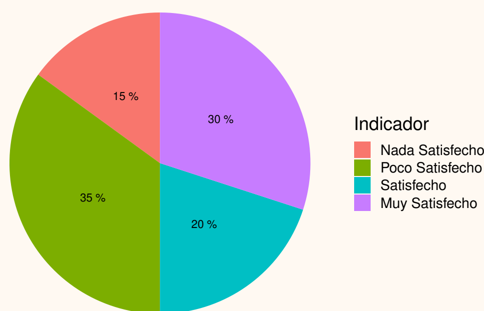


Figura 1: Grafico Likert

Por último **plotrix** con referencia `\ref{fig:fig4}`, al igual que el anterior usar `echo=FALSE` para omitir codigo.

Entrada 21

```
«fig4,out.width='70%',fig.cap='Grafico Likert',fig.pos='!hbt'»=
color <- c("#526189","#f6f6f6","#d3d9e7","#a3acc5")
#--------
fan.plot(x = tab$Freq,labels=as.character(tab$x3),
         col = color,ticks = 30)
@
```

Genera 21

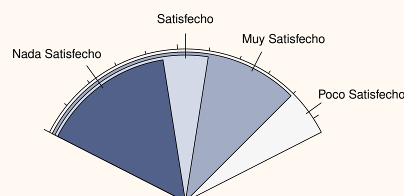


Figura 2: Grafico Likert

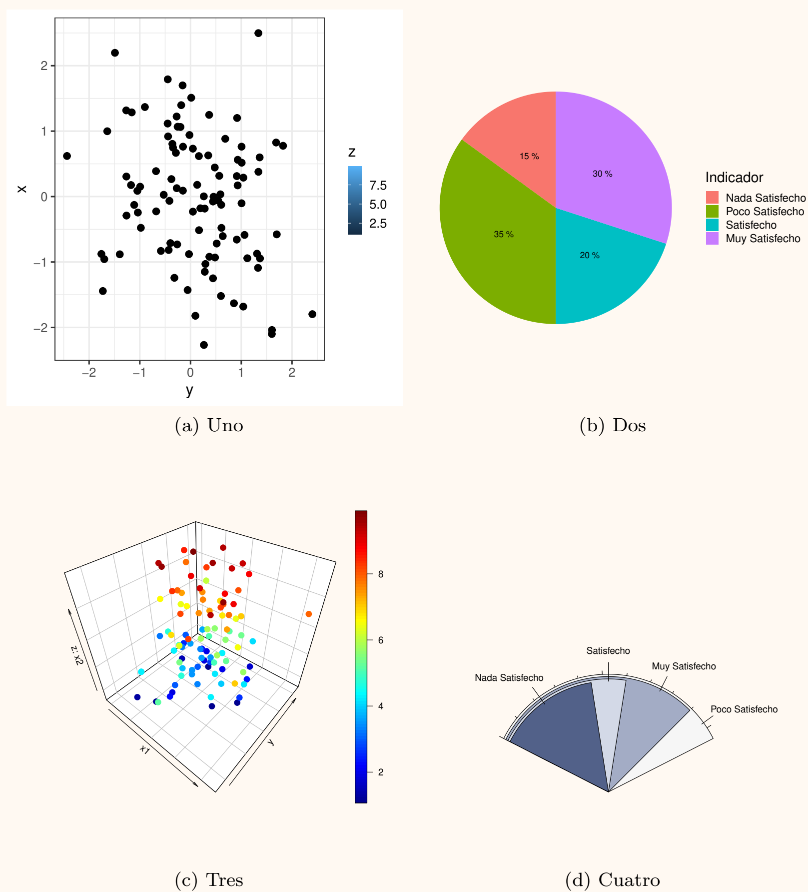
He aquí un ejemplo usando **fig.ncol** para unir gráficos en cuatro espacios y en cuanto a **fig.subcap** también debe coincidir con la cantidad de figuras. La librería **ggplot2** permite almacenar el gráfico en objetos, por ende **plotrix** y **plot3D** se crearan manualmente

Importante:
Cuando se agrega argumentos en **<< >>=** trate usted de hacerlo en una sola linea continua, no use saltos de lineas.

Entrada 22

```
%---------------
« general,echo=F,fig.show="hold", out.width="0.48\\textwidth",
  fig.cap="Todos los graficos",fig.subcap=c("Uno","Dos",
    "Tres","Cuatro"),fig.ncol=2,fig.pos='!hbt' »=
grafico1
grafico3
#--------------------------------
scatter3D(x=tb$x,y=tb$y,z=tb$z,cex=1.5,bty="b2",xlab="x1",
          ylab="y",zlab="z: x2",pch=16,col.panel="#99A69B")
#--------------------------------
fan.plot(x = tab$Freq,labels=as.character(tab$x3),
         col = color,ticks = 30)
@
```

Genera 22



(a) Uno

(b) Dos

(c) Tres

(d) Cuatro

Figura 3: Todos los graficos

Al igual que las demás, la referencia es **\ref{fig:general}**

5.3.5 Bibliografías

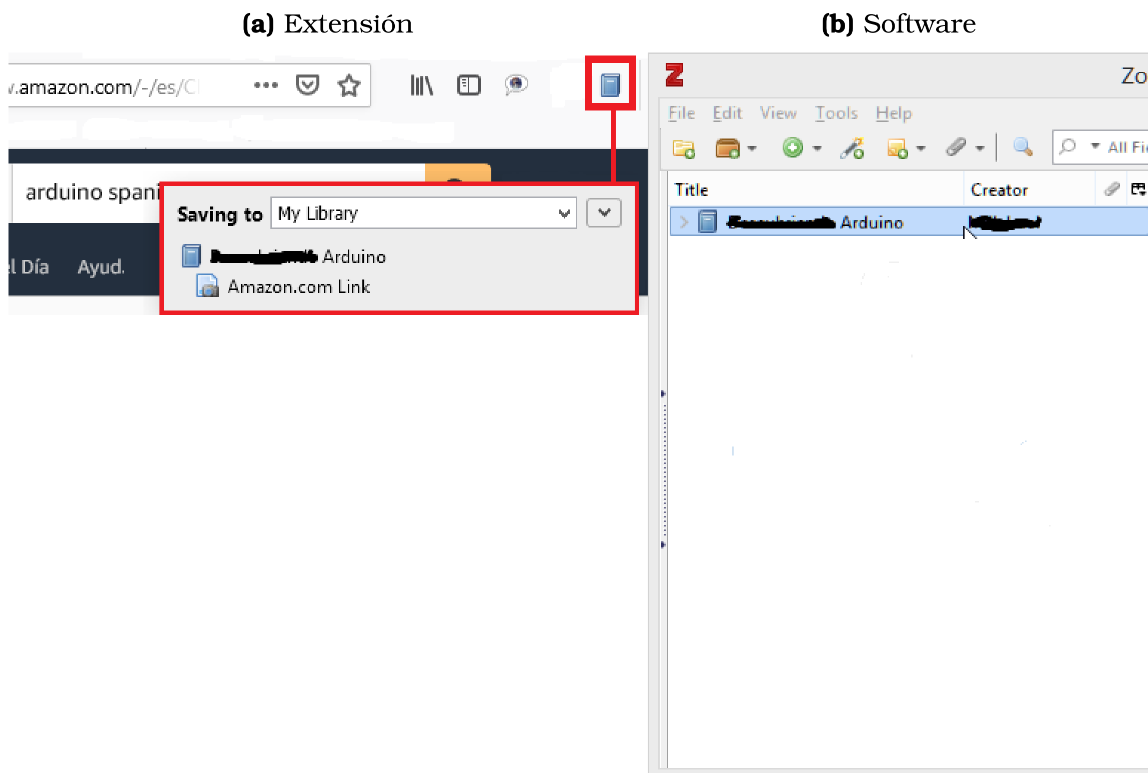
Muchos de los usuarios de Rstudio afirman que no compila la bibliografía y que para remediarlo usan otros softwares como es el caso de Texstudio,Texmaker,Kile,TexitEasy y TeXnicCenter .

La bibliografía es almacenada en un fichero con extensión **.bib** y al ser compilado con cualquier IDE LaTeXgenera otros ficheros como es el caso de **.bbl** que seria de gran importancia para el documento.

Existe softwares que permiten escribir la bibliografía, linkografia y otros de manera sencilla. Esta el Jabref(350MB) que se escribe de manera manual y el Zotero(85MB) que es más interactivo ya que posee una extensión para navegadores web como google chrome y mozilla.

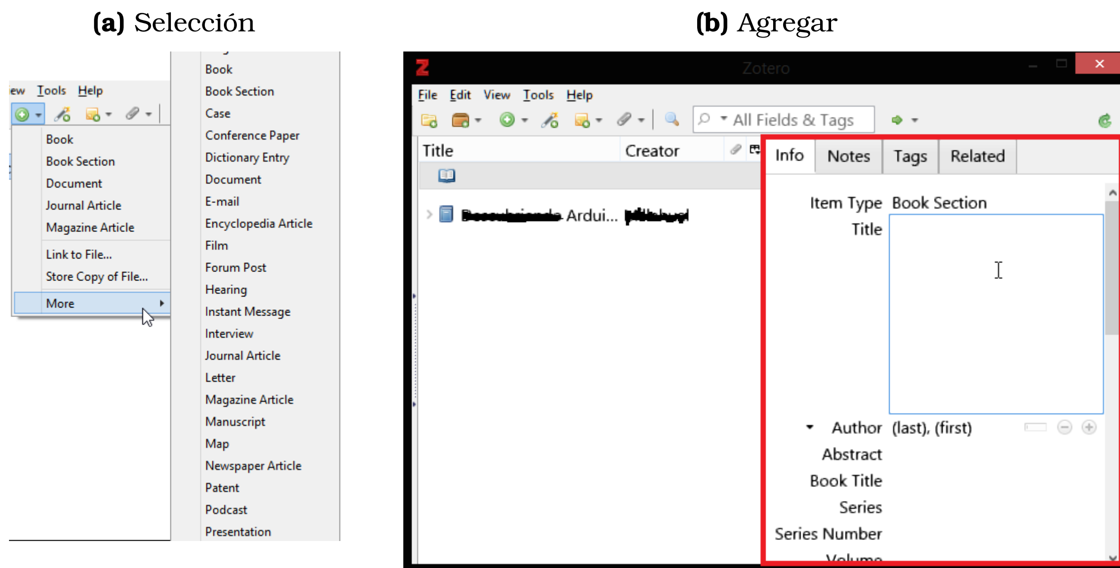
Para descargar zotero ingrese a su página web **https://www.zotero.org/** e instale, a continuación en su navegador web (mozilla o chrome) instale la extensión. ACLARACIÓN: PRIMERO EJECUTE LA APLICACIÓN ZOTERO Y DESPUÉS LA EXTENSIÓN (AMBOS SON DEPENDIENTES).

Figura 9.12: *La extensión y conexión con zotero*



Al hacer clic en el icono de la extensión zotero en el navegador aparecerá como se muestra en la figura 9.11a y al mismo tiempo también en el software figura 9.11b. No todas las páginas web permiten recopilar información zotero y es ahí donde se tiene que hacerlo manualmente

Figura 9.13: *Creando Nueva referencia desde el software Zotero*

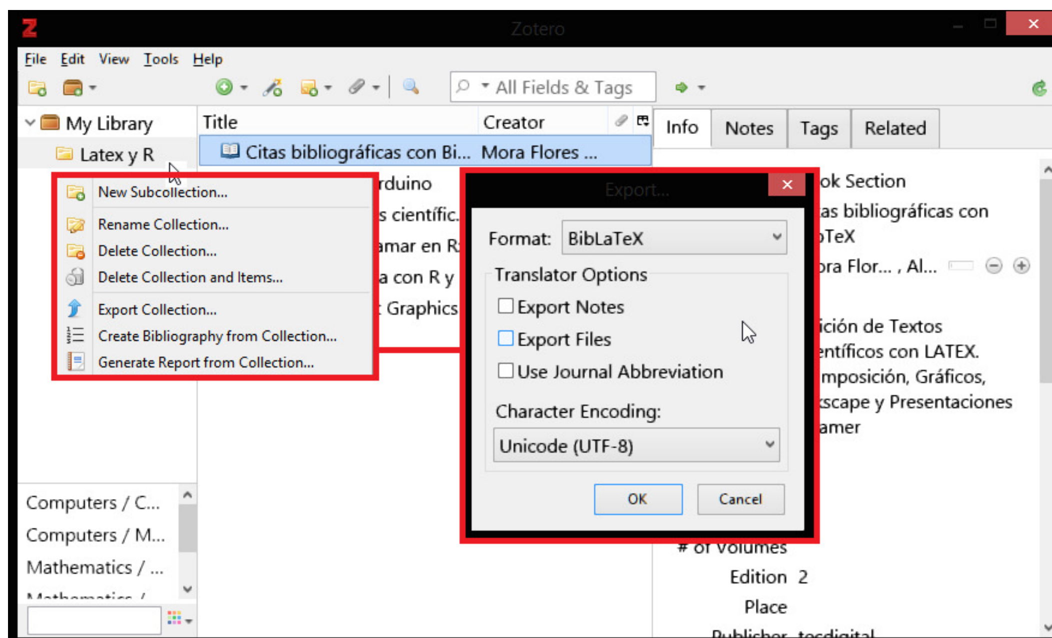


(a) Selección

(b) Agregar

Una vez que tengamos todas las referencias a nuestra disposición, se procederá a exportarlo a un fichero con un formato **BibLatex**

Figura 9.14: *Exportar desde Zotero*



Cuando hace clic derecho en la carpeta o cualquier referencia, aparecerá un submenu similar al cuadro rojo que esta a la izquierda y en ella has clic en **Export...** para que nuevamente aparezca una pequeña ventana donde debes elegir siempre la codificación en “UTF-8(Unicode)”.

Una vez hecho lo anterior, clic en **Ok** para guarda el fichero. Intente abrir ese fichero “bibliografia.bib” con notepad++ o cualquier otro editor de texto para editar lo siguiente

bibliografia.bib

```
@book{latex_2004_companion,
        title = {The {LaTeX} Companion},
        isbn = {978-0-13-338764-3},
        abstract = {This is the digial version of the printed ...},
        pagetotal = {1121},
        publisher = {Addison-Wesley Professional},
        author = {Mittelbach, Frank and Goossens, Michel and Bra...},
        date = {2004-04-23},
        langid = {english},
        note = {Google-Books-{ID}: {iX}9MAQAAQBAJ},
        keywords = {Computers / Electronic Publishing}
}
...
...
```

Tiene que editar **latex_2004_companion** por otro mas simple, esta etiqueta es muy importante porque con ese nombre vas a usar en

\citep{latex_2004_companion} (se ve muy enorme). Usaré uno mas corto **latex.2004**

Incluir paquete

Incluimos este paquete llamado "**biblatex**" en el documento actual, **es preferible que se encuentre entre los últimos paquetes**.

```
\documentclass[11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[spanish]{babel}
.....
.....
\usepackage{csquotes} %----<--<-------------incluir
\usepackage[style=apa]{biblatex} %<---------incluir
%--------------------------------
```

De forma predeterminada es el inglés y se necesita modificar el lenguaje al español

```
\DeclareLanguageMapping{spanish}{spanish-apa}
```

Aquí es donde tienes que colocar la dirección del **fichero bibliografia.bib** que habías exportado de zotero.

```
\addbibresource{bibliografia.bib}
```

Intentemos hacer una referencia sin modificar algún elemento:

Entrada 23

```
\begin{document}
..............
Según \parencite[326-329]{latex.2004} recomienda ...
..............
\printbibliography %<-----importante

\end{document}
```

Genera 23

Según (Mittelbach y col., 2004, pp. 326-329) recomienda .....

Como se observa se usa estas palabras "**y col , pp**". Si deseas hacer cambios en ella entonces presta atención a lo siguiente.

Entrada 24

```
%---------- modificar--------
\setcounter{smartand}{0}
\DefineBibliographyStrings{spanish}{%
    andothers = {et al. ,},
    pages     = {pag. },
}
%----------------------------

\begin{document}
..............
Según \parencite[326-329]{latex.2004} recomienda ...
..............
\printbibliography %<-----importante

\end{document}
```

Genera 24

Según (Mittelbach et al. , 2004, pag. 326-329) recomienda .....

Aquí la estructura general del fichero **documento.Rnw**

```
\documentclass[a4paper,12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[spanish]{babel}
\usepackage{graphicx,wrapfig}
\usepackage{microtype,float}
\usepackage[x11names,table]{xcolor}
\usepackage{colortbl,makecell}

%---------> entorno matemático
\usepackage{amsmath,amssymb,array,amsfonts,latexsym,amssymb}
\usepackage{cancel} % tachado de expresiones
\usepackage{textcomp,wasysym,longtable}
\let\Square\relax % importante
\usepackage{bbding}
\usepackage{pdflscape,tabu}

%----------> entorno tablas
\usepackage{threeparttable,threeparttablex}
\usepackage{multicol,multirow,booktabs}
\usepackage[capposition=top]{floatrow} % título arriba
\usepackage{multicol,multirow,booktabs}
\usepackage{threeparttable}
\usepackage{longtable}
%------------- ----
\usepackage{csquotes}
\usepackage[style=apa]{biblatex}
\DeclareLanguageMapping{spanish}{spanish-apa}
\addbibresource{bibliografia.bib}  %<----- dirección fichero bib
%---------- modificar--------
\setcounter{smartand}{0}
\DefineBibliographyStrings{spanish}{%
andothers = {et al. ,},
pages = {pag. },
}
%-------------> vinculos
\usepackage{hyperref}

%--------> crear comandos <-----
\newcolumntype{a}{>{\columncolor{Brown1}}c}
\newcolumntype{e}{>{\columncolor{DarkOrange1}}c}
```

```
%----------------
\newcommand{\tex}[1]{\textcolor{white}{\textbf{#1}}}
%--------------------------------

%====================================================
\title{Titulo del articulo}
\author{el autor}
\date{}% fecha  formato: dia/mes/año
%====================================================

% --> R librerias
<<echo=F,warning=FALSE,message=FALSE>>=
library(knitr)
library(kableExtra)
library(xtable)
options(xtable.comment = FALSE)
library(ggplot2)
library(plotrix)
@


\begin{document}
\frontmatter

\maketitle
\tableofcontents

\mainmatter
%======================= de aqui inicia
\section{H1}
\subsection{H2}
\subsubsection{H3}

%======================= aqui termina
\begin{appendix}
  \listoffigures
  \listoftables
\end{appendix}

\nocite{*} % imprime todo el contenido de .bib
```

```
\printbibliography[
heading=bibintoc,
title={Bibliografía}
]
\end{document}
```

CAPÍTULO 10

Referencias

1.1 Bibliografía

[1] Walter Mora F Alexánder Borbón A. *Edición de textos científicos con LaTeX. Composición, gráficos, diseño editorial y presentaciones beamer*. 2nda. Revista digital Matemática Educación e Internet, 2017. ISBN: 978-9977-66-227-5. URL: **https://tecdigital.tec.ac.cr/revistamatematica/Libros/**.

[2] Winston Chang. *R Graphics Cookbook: Practical Recipes for Visualizing Data*. Google-Books-ID: fxL4tu5bzAAC. .º'Reilly Media, Inc.", 6 de dic. de 2012. 414 págs. ISBN: 978-1-4493-6310-9.

[3] Dilip Datta. *LaTeX in 24 Hours: A Practical Guide for Scientific Writing*. Springer International Publishing, 2017. ISBN: 978-3-319-47830-2. DOI: **10.1007/978-3-319-47831-9**. URL: **https://www.springer.com/gp/book/9783319478302**.

[4] Colin Gillespie y Robin Lovelace. *Efficient R Programming: A Practical Guide to Smarter Programming*. Google-Books-ID: 8kWvDQAAQBAJ. .º'Reilly Media, Inc.", 8 de dic. de 2016. 220 págs. ISBN: 978-1-4919-5075-3.

[5] Garrett Grolemund. *Hands-On Programming with R: Write Your Own Functions and Simulations*. Google-Books-ID: S04BBAAAQBAJ. .º'Reilly Media, Inc.", 13 de jun. de 2014. 249 págs. ISBN: 978-1-4493-5911-9.

[6] Nicholas J. Horton y Ken Kleinman. *Using R and RStudio for Data Management, Statistical Analysis, and Graphics*. Google-Books-ID: W1G3BgAAQBAJ. CRC Press, 10 de mar. de 2015. 280 págs. ISBN: 978-1-4822-3737-5.

[7] Robert Kabacoff. *R in Action: Data Analysis and Graphics with R*. Google-Books-ID: aVnanAEACAAJ. Manning, 2015. 579 págs. ISBN: 978-1-61729-138-8.

[8] Michael Lawrence y John Verzani. *Programming Graphical User Interfaces in R*. Google-Books-ID: VeKEDwAAQBAJ. CRC Press, 14 de dic. de 2018. 481 págs. ISBN: 978-1-315-36051-5.

[9] Mark P. J. Van der Loo. *Learning RStudio for R Statistical Computing*. Google-Books-ID: EE8M9HCJok4C. Packt Publishing Ltd, 1 de ene. de 2012. 206 págs. ISBN: 978-1-78216-061-8.

[10] Luis Rández. *Introducción a LATEX*. 1ra. Universidad de Zaragoza, 2019. URL: **http://pcmap.unizar.es/~pilar/latex.pdf**.

[11] Donato Teutonico. *ggplot2 Essentials*. Google-Books-ID: zN0DCgAAQBAJ. Packt Publishing Ltd, 25 de jun. de 2015. 234 págs. ISBN: 978-1-78528-755-8.

[12] John Verzani. *Getting Started with RStudio*. Google-Books-ID: q95Nyozjda4C. .O'Reilly Media, Inc.", 23 de sep. de 2011. 97 págs. ISBN: 978-1-4493-0903-9.

Índice de cuadros

